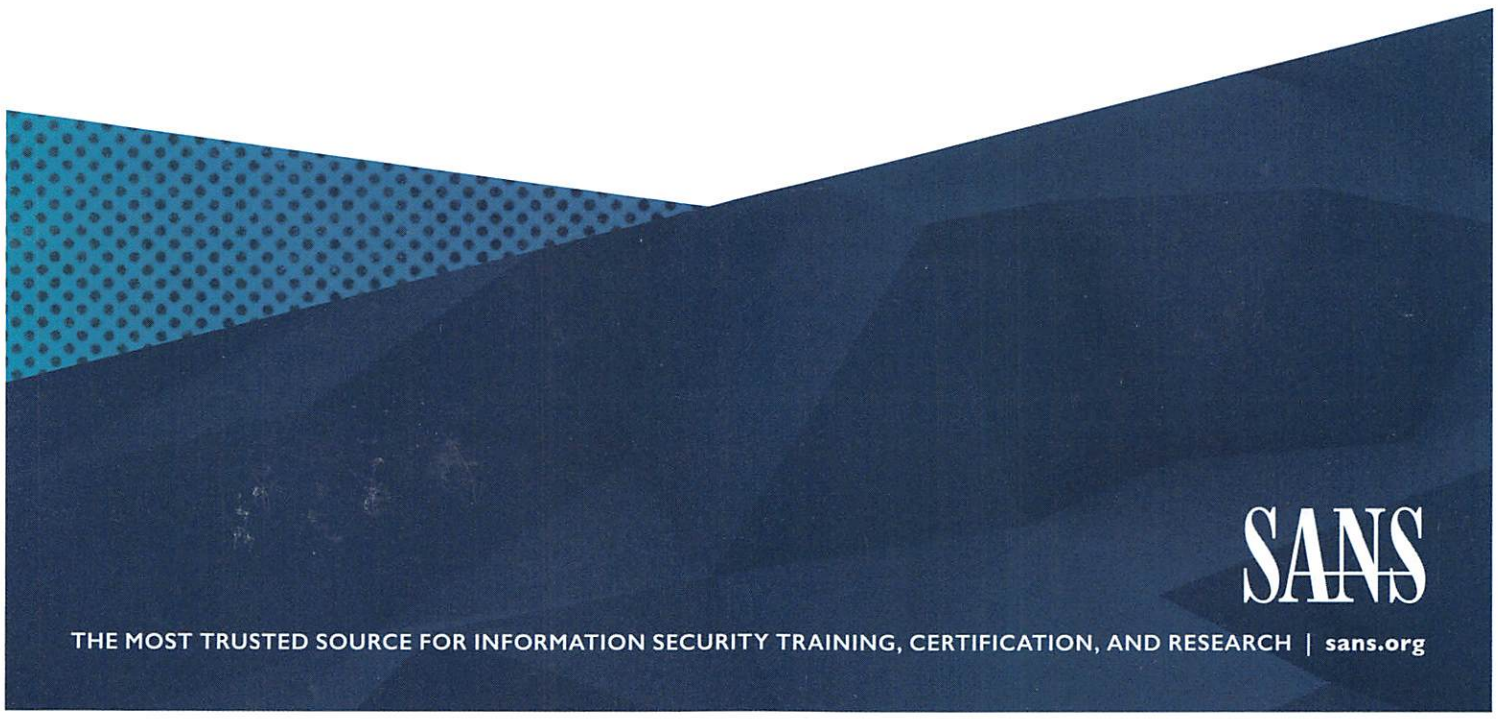
518.4
Application Data Analysis



SANS

518.4
Application Data Analysis

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

Copyright © 2020, Sarah Edwards. All rights reserved to Sarah Edwards and/or SANS Institute.

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

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 marks 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.

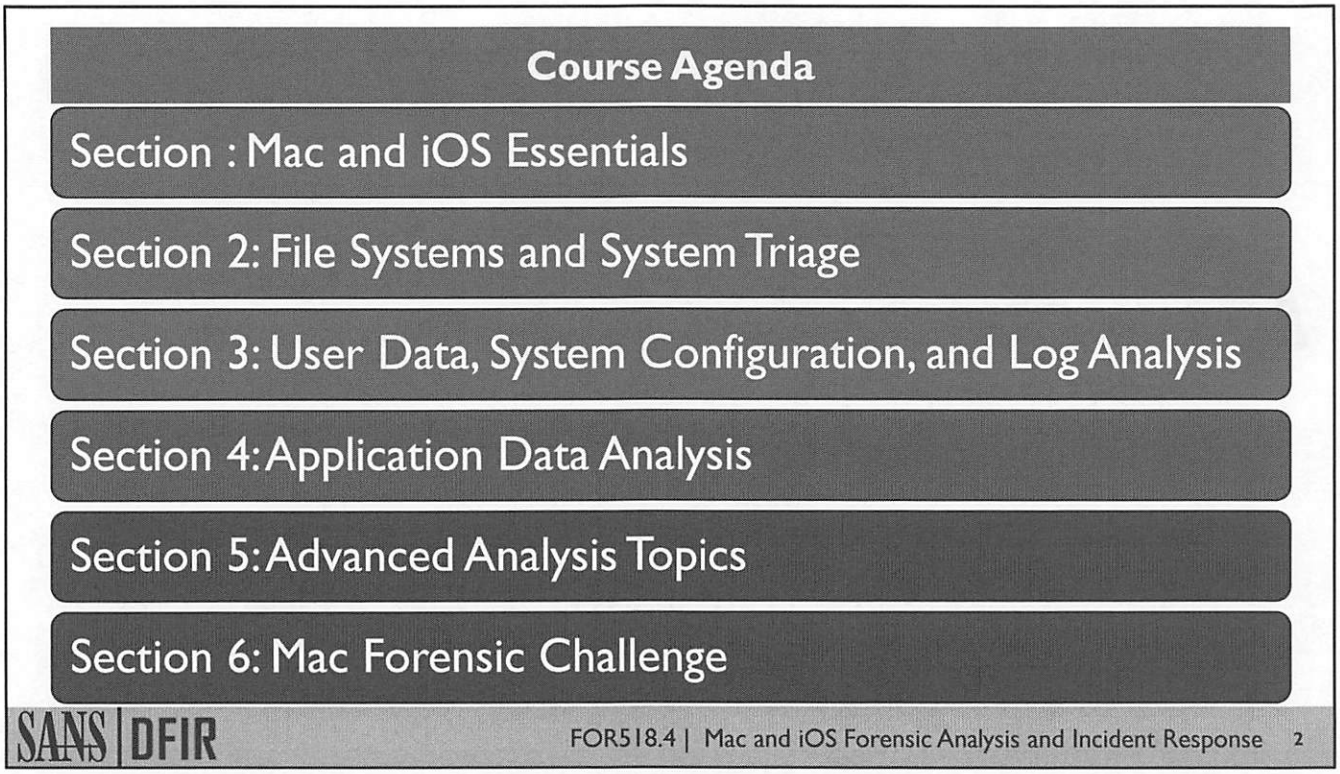
FOR518.4 | Mac and iOS Forensic Analysis and Incident Response

FOR518 Section 4: Application Data Analysis

© 2020 Sarah Edwards | All Rights Reserved | Version F01_01

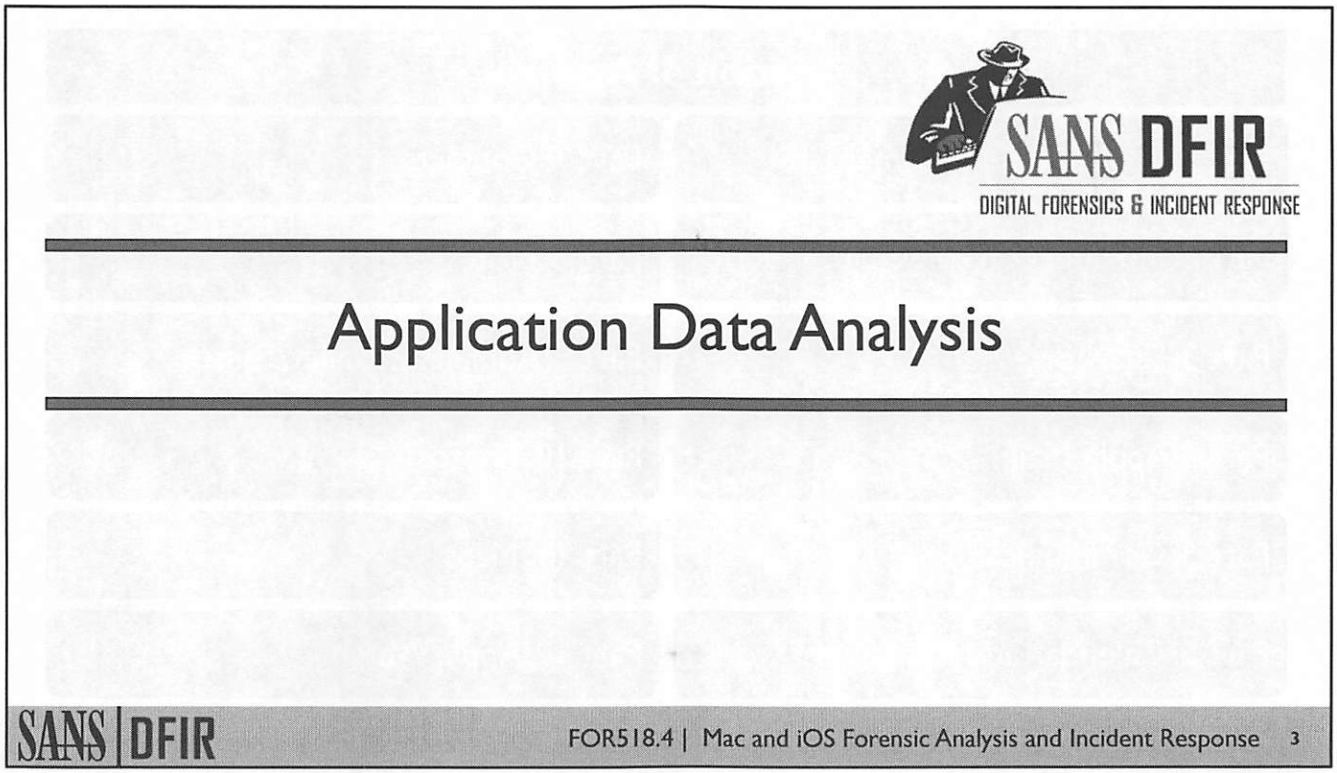
Author: Sarah Edwards
oompa@csh.rit.edu
mac4n6.com
http://twitter.com/iamevltwin

https://digital-forensics.sans.org/
http://twitter.com/sansforensics



This page intentionally left blank.

© 2020 Sarah Edwards



This page intentionally left blank.

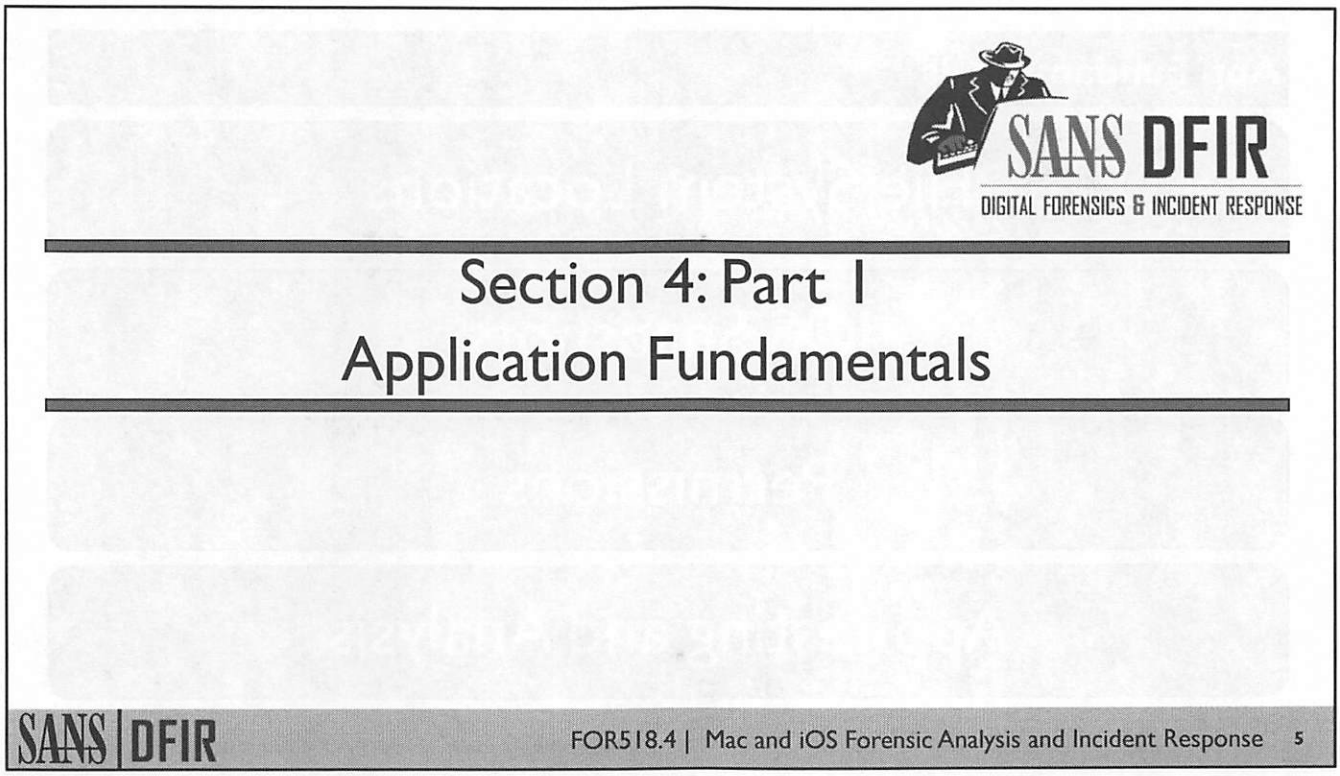
© 2020 Sarah Edwards

Section 4: Agenda

- Part 1: Application Fundamentals
- Part 2: Introduction to SQLite Queries
- Part 3: Safari Browser
- Part 4: Apple Mail
- Part 5: Communication
- Part 6: Calendar and Reminders
- Part 7: Contacts
- Part 8: Notes
- Part 9: Apple Pay, Wallet, Passes
- Part 10: Photos
- Part 11: Maps
- Part 12: Apple Watch

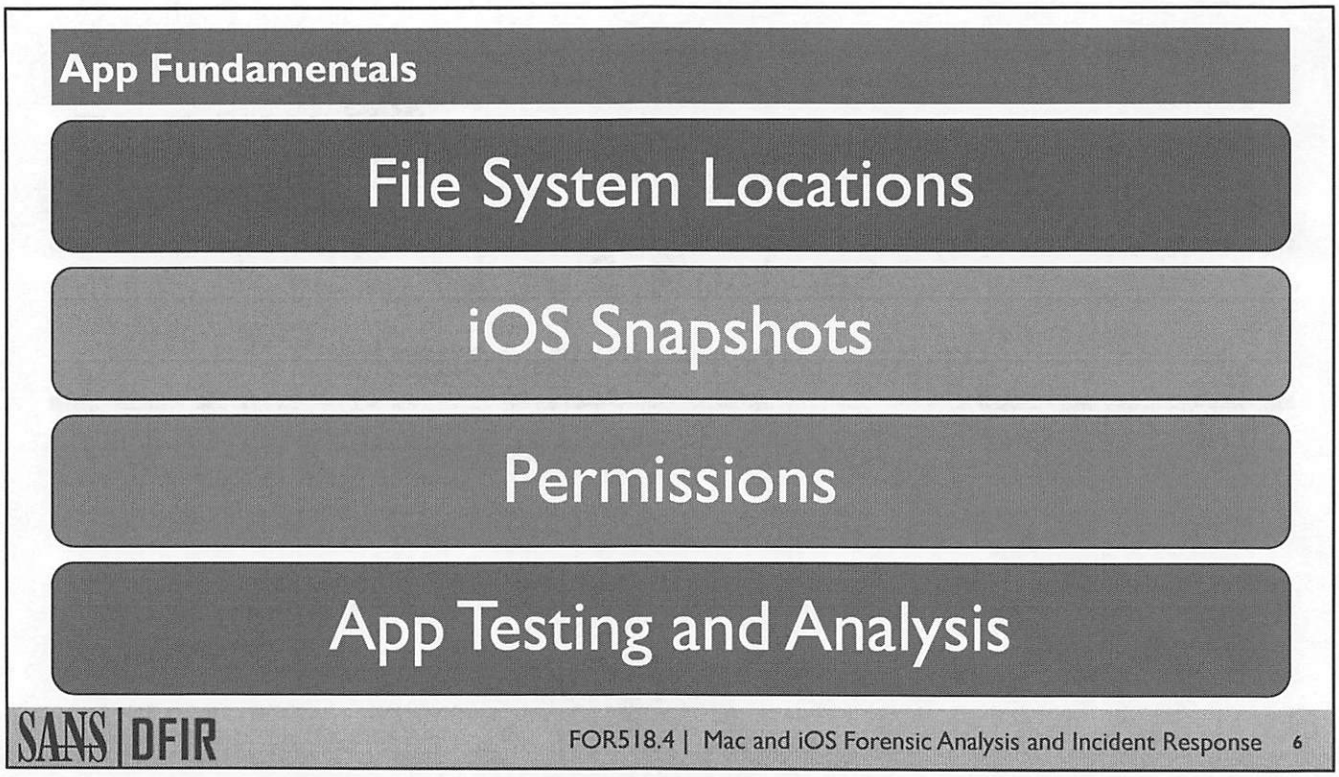
This page intentionally left blank.

© 2020 Sarah Edwards



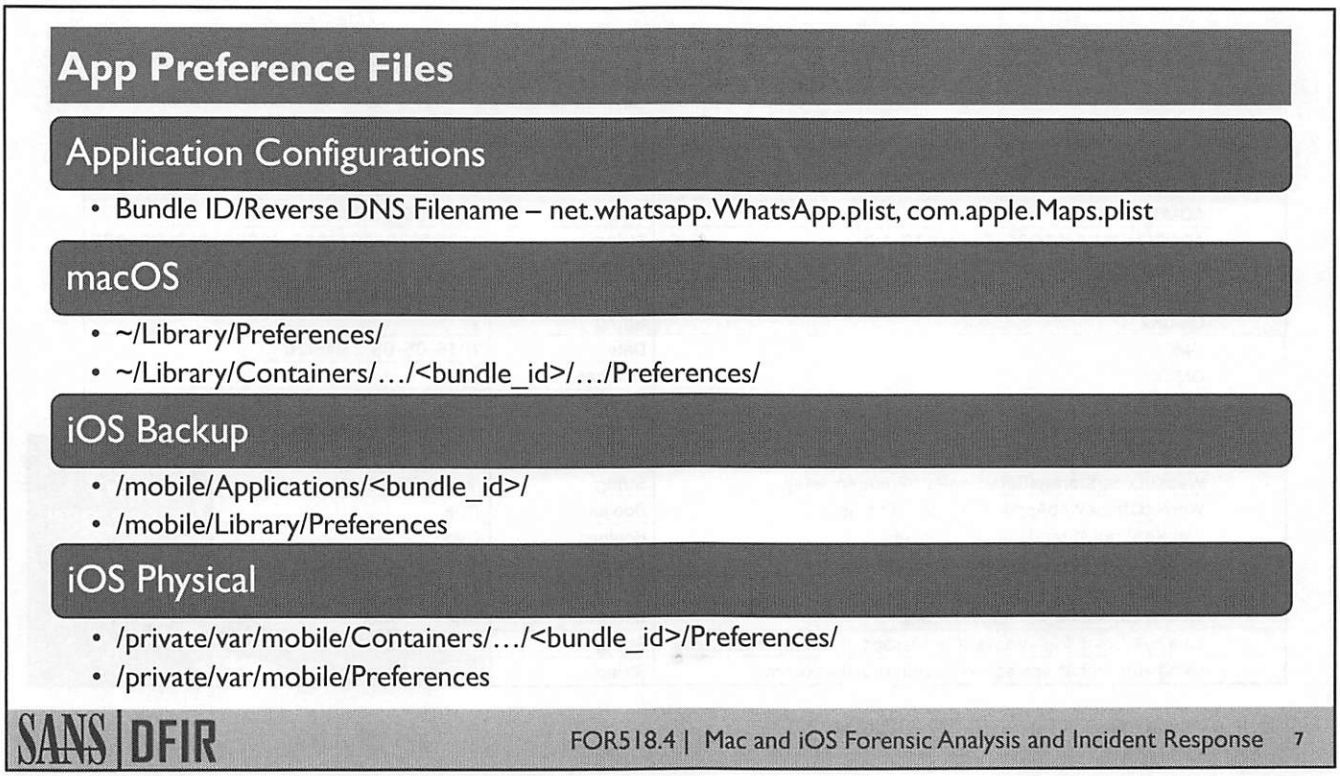
This page intentionally left blank.

© 2020 Sarah Edwards



This page intentionally left blank.

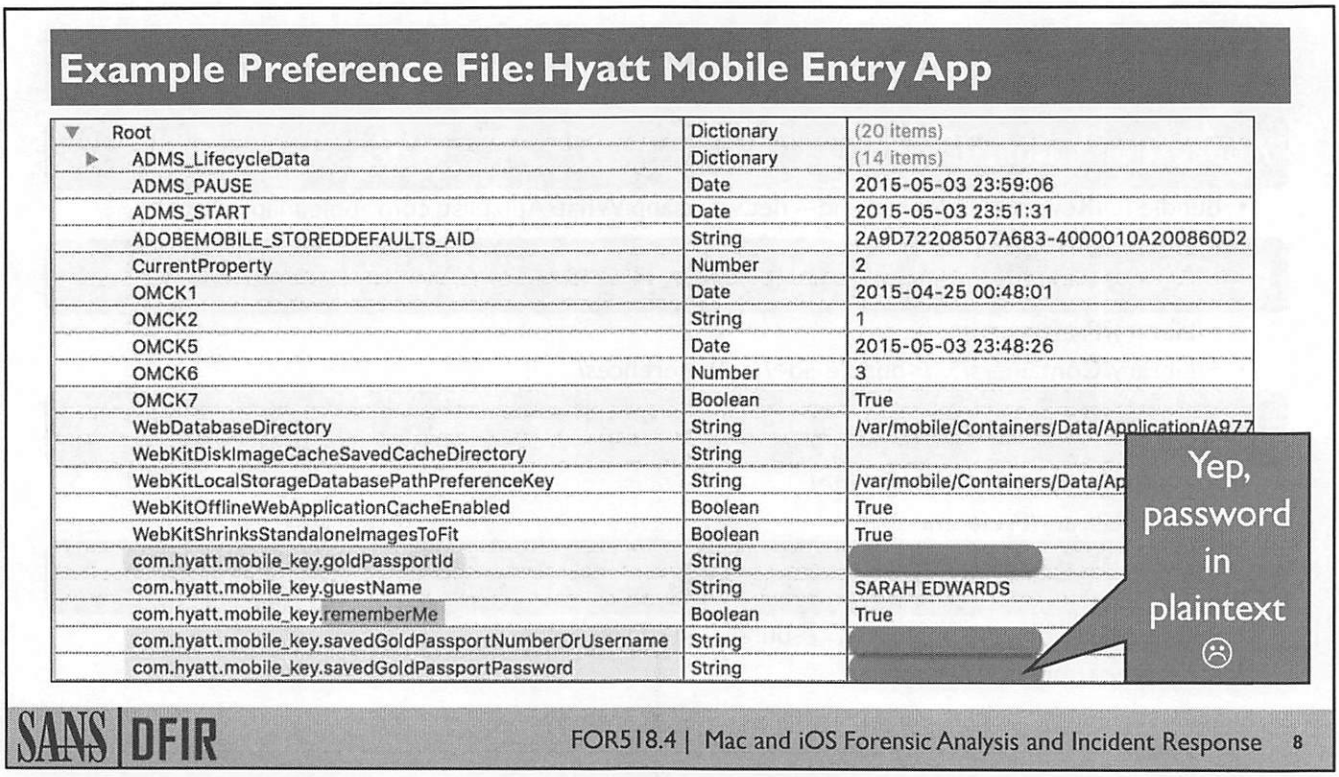
© 2020 Sarah Edwards



Preference files are usually plist files that contain configuration data for a particular application. This data can include a wide variety of items.

These preference files are plist files that are named using the bundle ID, the reverse DNS format.

© 2020 Sarah Edwards



In addition to application data such as messages or contact lists, an analyst may need to look for user IDs such as email addresses and account numbers. Analysts may also find passwords in plaintext associated with various accounts.

The example above is from the Hyatt Mobile Entry application (`com.hyatt.MobileEntry.plist`). The app configuration plist files may contain some interesting information and are always worth quickly reviewing. If an analyst is trying to find passwords used to attempt to break into other encrypted containers, seemingly random applications on that user's device may contain this information in an easy-to-get way! Why brute force when you can get passwords in plaintext!

© 2020 Sarah Edwards

App Databases and Other Files

SQLite Databases, Proprietary Files, Logs – Anything Goes!

macOS

- ~/Library/
- ~/Library/Application Support/
- ~/Library/Containers/…

iOS Backup

- /mobile/Applications/<bundle_id>/
- /mobile/Library/

iOS Physical

- /private/var/mobile/Containers/… /<bundle_id>/Library/
- /private/var/mobile/Library/

Databases and other application files can be found in these locations. Each application will have its own files representing the data used for the apps themselves. Common files seen are SQLite database files, proprietary data, and maybe some log files.

© 2020 Sarah Edwards

Application Caches

Cache.db SQLite Database and/or Other Cached Files

macOS

- ~/Library/Caches/
- ~/Library/Containers/…/<bundle_id>/…/Cache/

iOS Backup

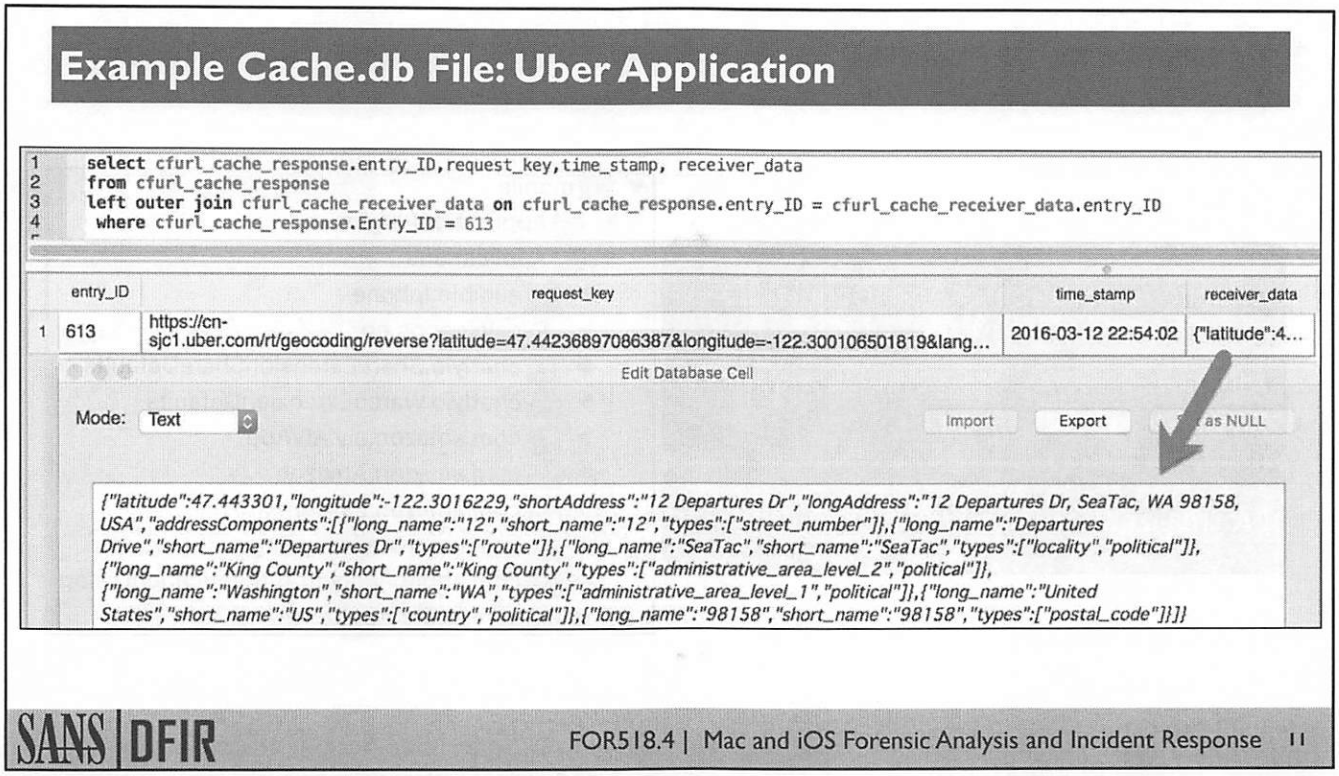
- Probably little to none; cached items rarely get backed up

iOS Physical

- /private/var/mobile/Containers/…/<bundle_id>/Library/Caches/
- /private/var/mobile/Library/Cache/<bundle_id>

Application caches are always worth a review. Browsers are often the most common for an analyst to review; however, other apps may retain cached data that could be of importance in an investigation.

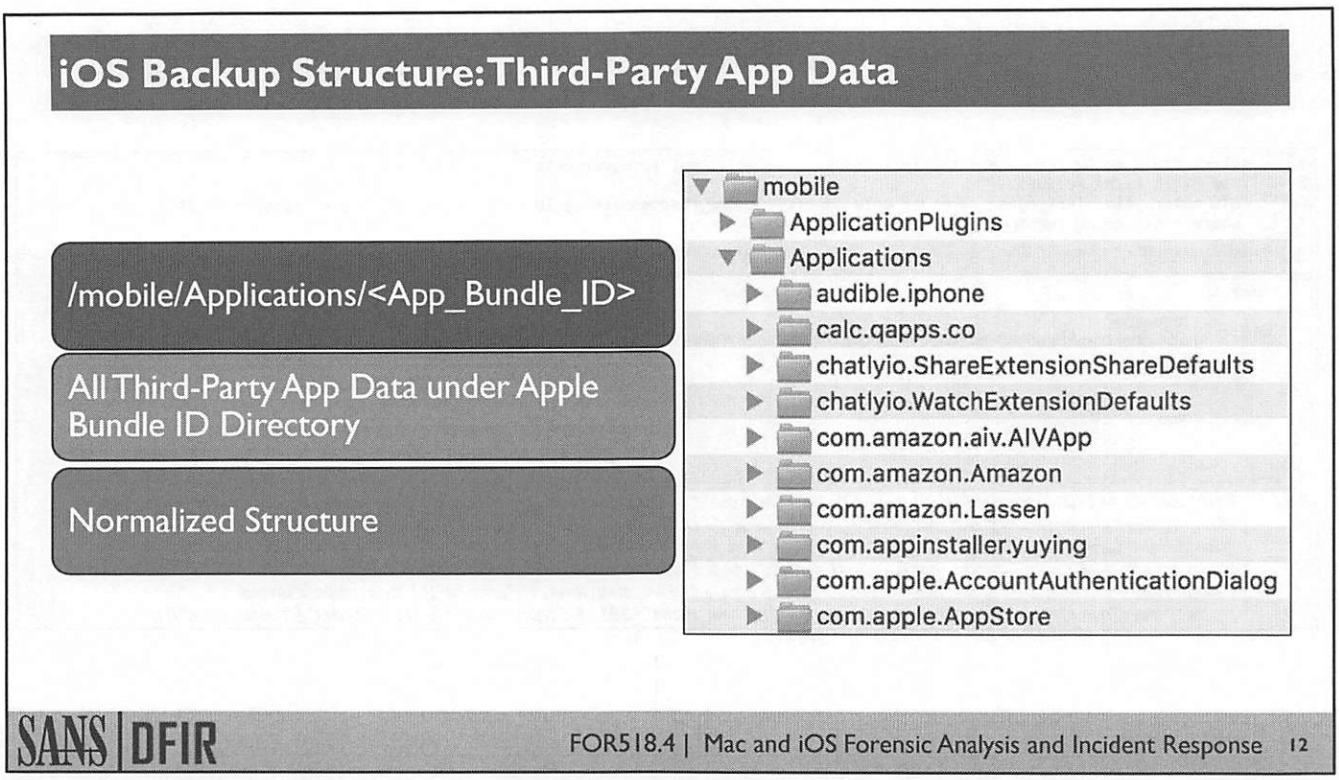
© 2020 Sarah Edwards



Another type of data an analyst may be searching for is location data. Depending on the acquisition type, the analyst may not have access to the location databases on iOS. An examiner can get creative and look in many types of applications for this data.

The example above shows the `Cache.db` (`com.ubercab.UberClient/Cache.db`) file for the Uber application. Any applications that use location services may be good candidates for location data. The Uber `cache.db` database contains HTTP URL queries containing location data for where the user was requesting a car. The response (shown in the "Edit Database Cell" window) contains the return information from this query, which also contains location information with more granular detail.

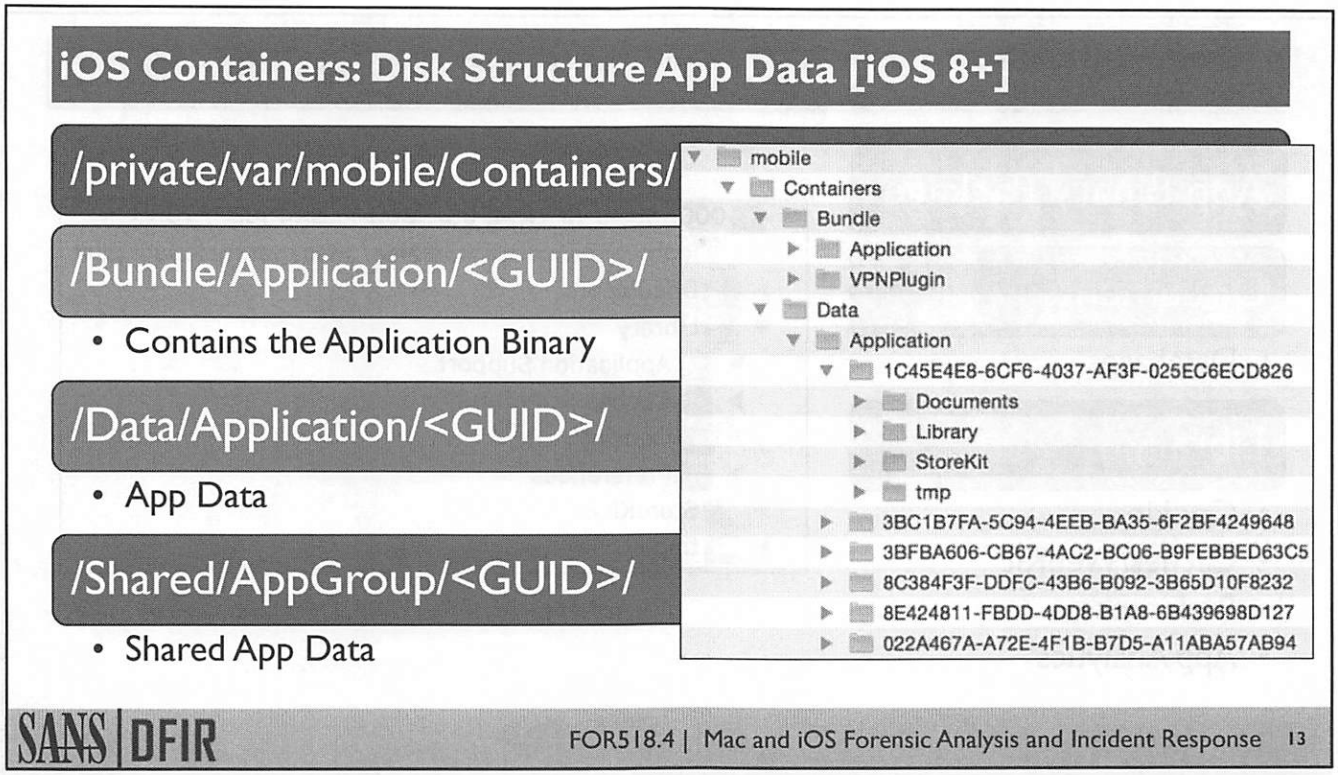
© 2020 Sarah Edwards



The iOS backup structure when shown in BlackLight (shown) and other tools will show a normalized data structure. This is not exactly how it is being stored on disk.

All application data is stored under the directory associated with that application's Bundle ID.

© 2020 Sarah Edwards



iOS 8 introduced Containers to the file system. Application data has been slightly moved around to fit this new sandboxing implementation.

Much of the same data (i.e., `/Documents`, `/Library/Caches`, `/Library`, etc.) is stored under a new file path, as shown above.

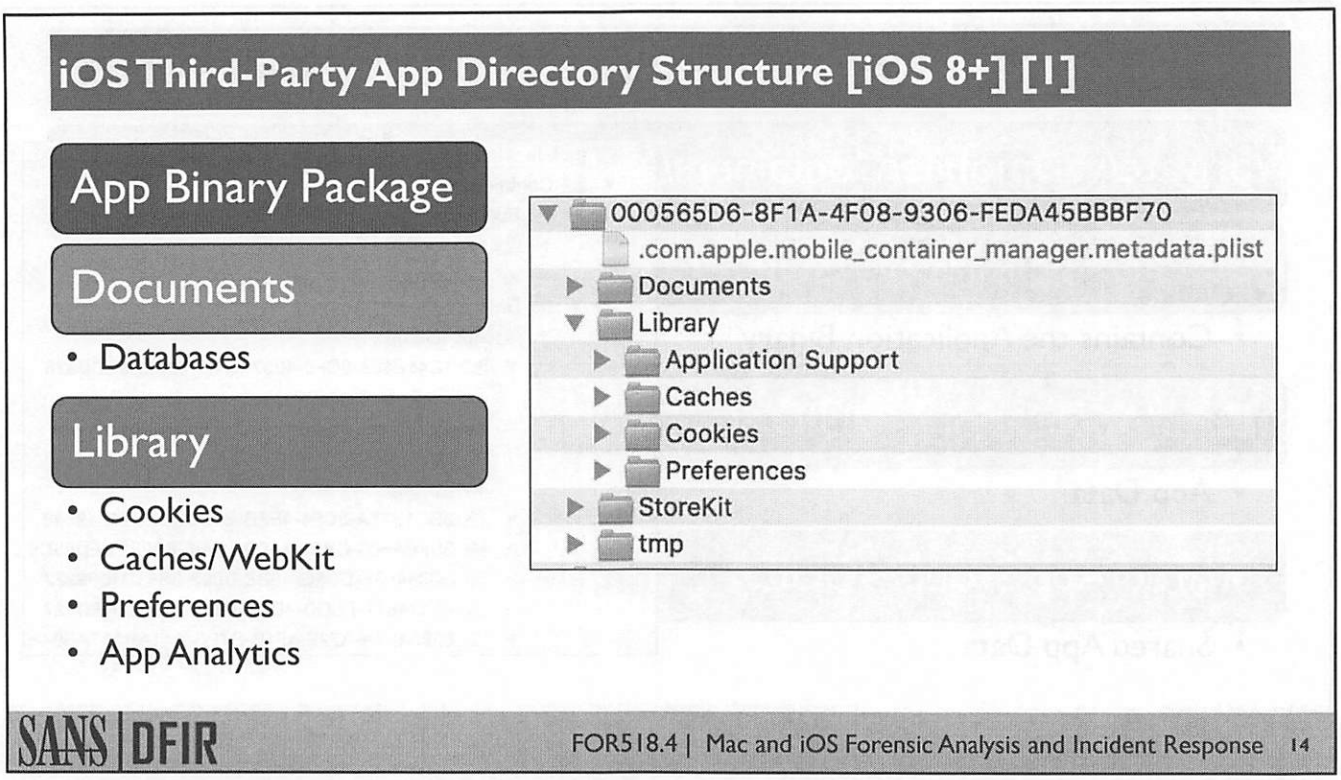
A database in iOS 7 might have had the path:
`/private/var/mobile/Applications/<GUID>/Documents/somedatabase.db`.

In iOS 8, this path may be
`/private/var/mobile/Containers/Data/Application/<GUID>/Documents/somedatabase.db`

Also new to iOS 8 is "shared" data. This can be data shared among apps by the same developer, such as the Microsoft Word, PowerPoint, and Excel applications (for iDevices of course!)

Reference:
`https://developer.apple.com/library/content/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html`

© 2020 Sarah Edwards



Each third-party application directory contains a standard set of folders which (usually) has a basic set of data contained within them.

The Documents directory may contain documents, but also application databases. The Library directory may contain cookie data, cached data in the cache.db databases, and configuration plists, as well as application analytics information.

For iOS 8 and newer devices, the Shared directory should also be reviewed for app data. Some apps will use this directory for all their files instead of the one above.

© 2020 Sarah Edwards

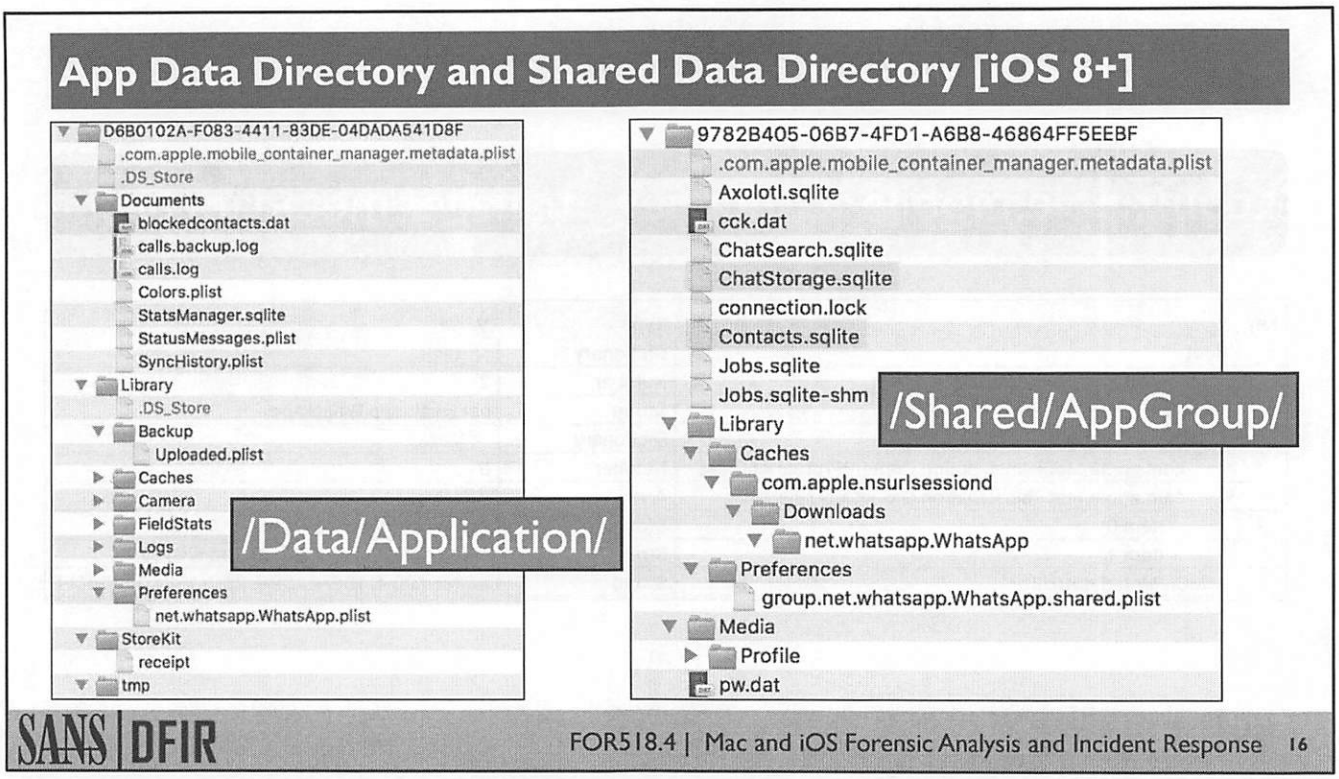
iOS Third-Party App Directory Structure [iOS 8+] [2]

.com.apple.mobile_container_manager.metadata.plist

Key	Type	Value
▼ Root	Dictionary	(4 items)
MCMMetadataContentClass	Number	2
MCMMetadataIdentifier	String	net.whatsapp.WhatsApp
▼ MCMMetadataInfo	Dictionary	(2 items)
com.apple.MobileInstallation.ContentProtectionClass	Number	0
▼ com.apple.MobileInstallation.GroupContainerIDs	Array	(2 items)
Item 0	String	group.com.facebook.family
Item 1	String	group.net.whatsapp.WhatsApp.shared
MCMMetadataUUID	String	3166F790-D188-4A09-B95C-1F84D8F886FF

Each application on iOS 8 or newer will have a hidden file in its root directory containing the Bundle ID of the application as well as other metadata, such as protection classes and group bundle ID information.

© 2020 Sarah Edwards



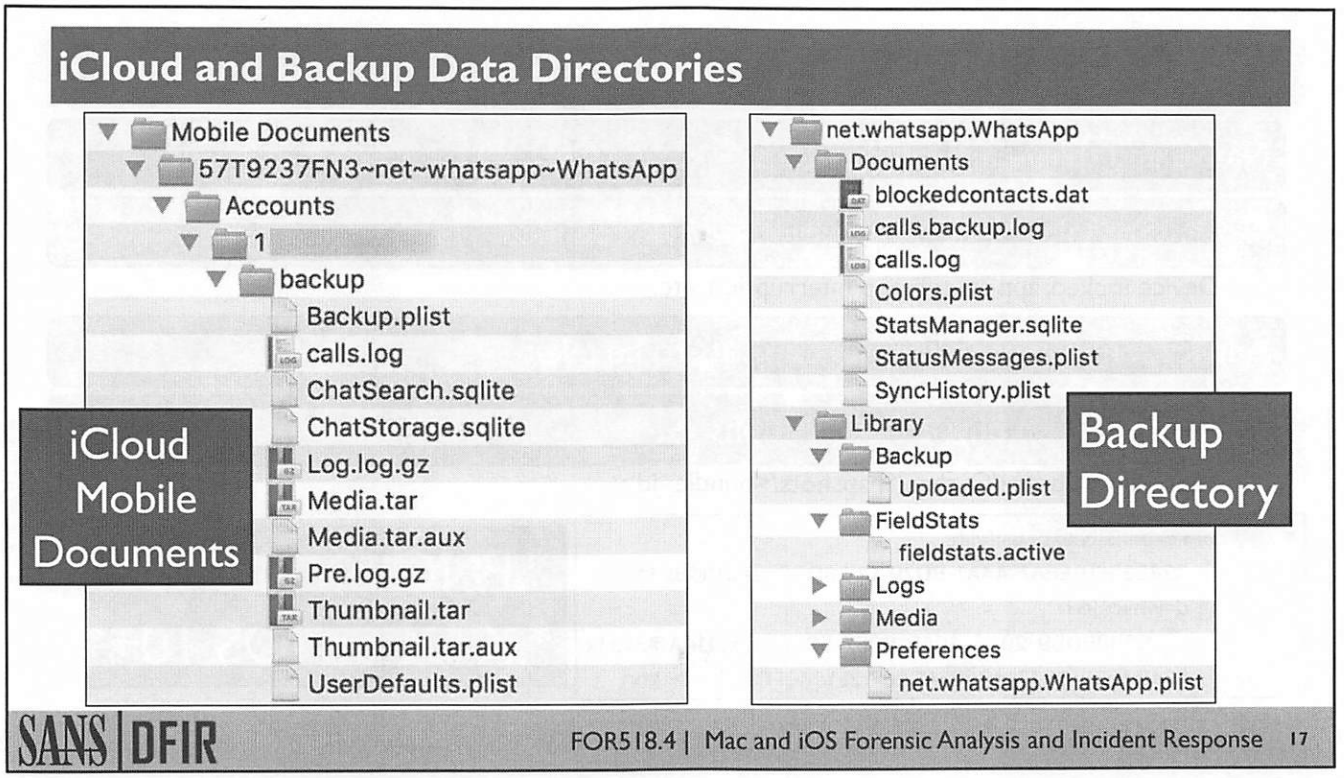
The differences between the `/Data/Application/` and the `/Shared/AppGroup/` directories in the app containers can be quite different and it's always worth looking at both to find the information you are seeking.

On the left is a screenshot of the `/Data/Application` directory for the WhatsApp app (`net.whatsapp.whatsapp`). This shows some information, but if you are looking for the Messages and Contacts databases for the WhatsApp app, you will have to look in the `/Shared/AppGroup/` directory instead.

Not shown in the screenshots above, the Media directory in the `/Data/Application/` directory contains the media (photos/videos) sent back and forth in conversations—this media is not in the `/Shared/AppGroup/` directory for WhatsApp.

These examples come from physical-logical dumps.

© 2020 Sarah Edwards



Along with the `/Data/Application/` and `/Shared/AppGroup/` directories, the analyst should also look at the iCloud Mobile Documents directory (shown left), as well as any backups that may be found on the device.

Each directory structure will choose to store only a specific directory. If the app does not want to back up anything to iCloud, you will likely not find much in the `Mobile Documents` directory. If the app developer chooses to back up specific items in local backups, you will find these in the normalized data structure, such as the example shown on the right for WhatsApp.

The `Mobile Documents` directory example comes from a physical-logical dump.

© 2020 Sarah Edwards

iOS Application Snapshots

Native and third-party applications

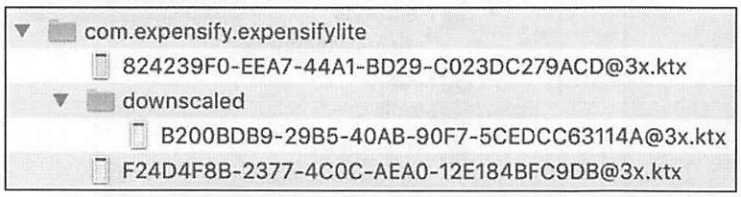
Screenshot taken when application is backgrounded

- Device locked, app switch, app interruption, etc.

Application may not allow screenshots to be taken

Only available on physical acquisitions

- <app_dir>/Library/Caches/Snapshots/<bundle_id>/



*.PNG on iOS 9-
*.KTX on iOS 10+

Whenever an application is backgrounded, a screenshot is taken of what the user is currently viewing on the screen and is saved to the file system. This allows for smoother transitions and the appearance of faster switching between applications.

A screenshot is taken when an app is switched to another application, when the device is locked, or when an application is interrupted by another application (i.e., receiving a phone call).

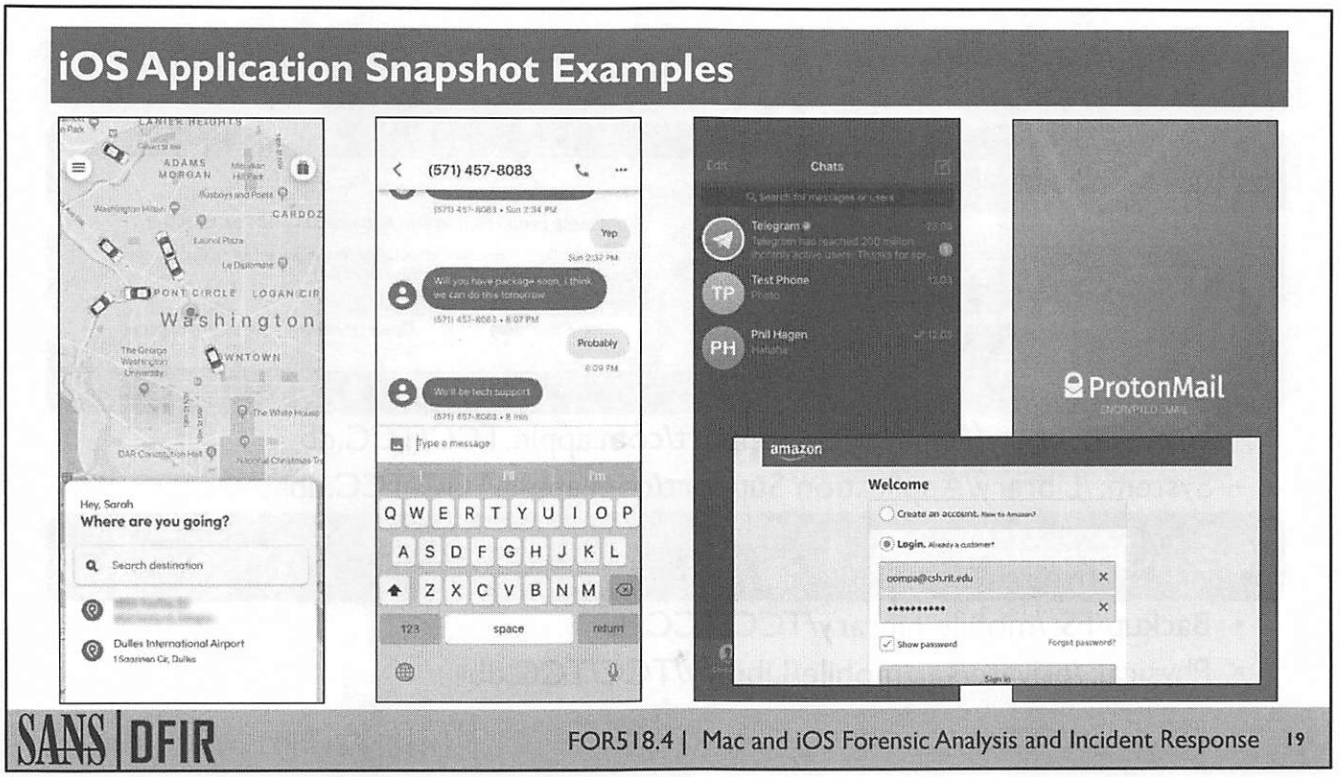
These screenshots may be available for native as well as third-party applications. Depending on the application, developers may choose to limit this screen capture by explicitly telling their code to do so. This may be done for "security", as sensitive data may be saved in these applications. Many messaging or banking applications do not use this feature, or they use a decoy image instead.

These screen captures are only available on physical acquisitions.

These screenshots are PNG files on iOS 9 and older. On iOS 10, the files use a newer format—KTX. These files can be viewed in the macOS Preview application.

Reference:
`https://www.khronos.org/opengles/sdk/tools/KTX/file_format_spec/`

© 2020 Sarah Edwards



Snapshots can be used in many ways to help an investigation. You are viewing a snapshot in time of exactly what the user was seeing. These pictures have the potential to contain information that may not necessarily be stored in the applications database or plist files.

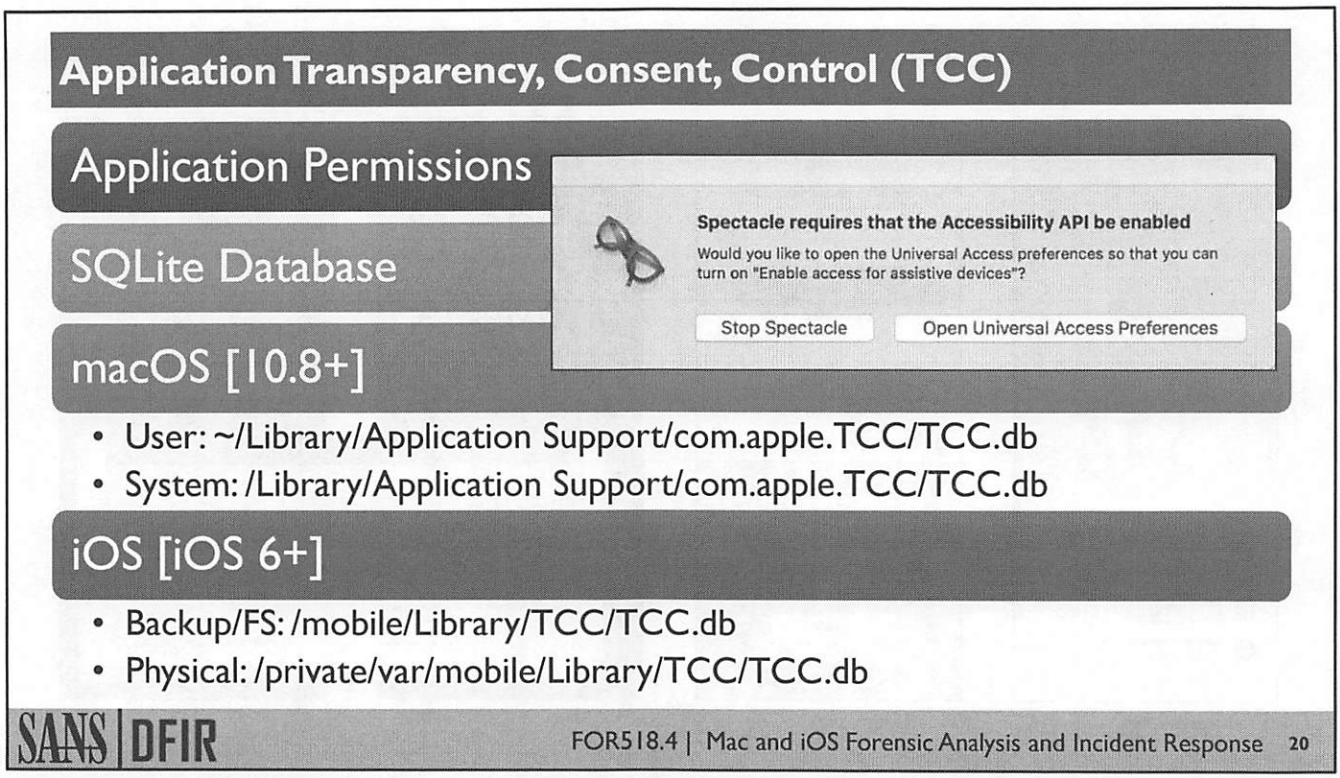
Snapshots may be different for applications depending whether or not the user was using the application in portrait or landscape modes.

Contacts / Address Book: Last contact viewed
Camera: Last "thing" viewed, perhaps without having an actual photo taken
Calendar: Schedule the last day viewed
Messages: List of messaged contacts with last message to them—could potentially be captured before message is deleted
Third-Party Messaging Apps: Messages viewed, typed
Mapping Applications: Last viewed location
Web Browsers: Last viewed webpage

The examples above from left to right are the following apps:

- Lyft
- Google Voice
- Telegram ("Secure" Messenger)
- Proton Mail – Secure Email
- Safari Browser (horizontal)

© 2020 Sarah Edwards

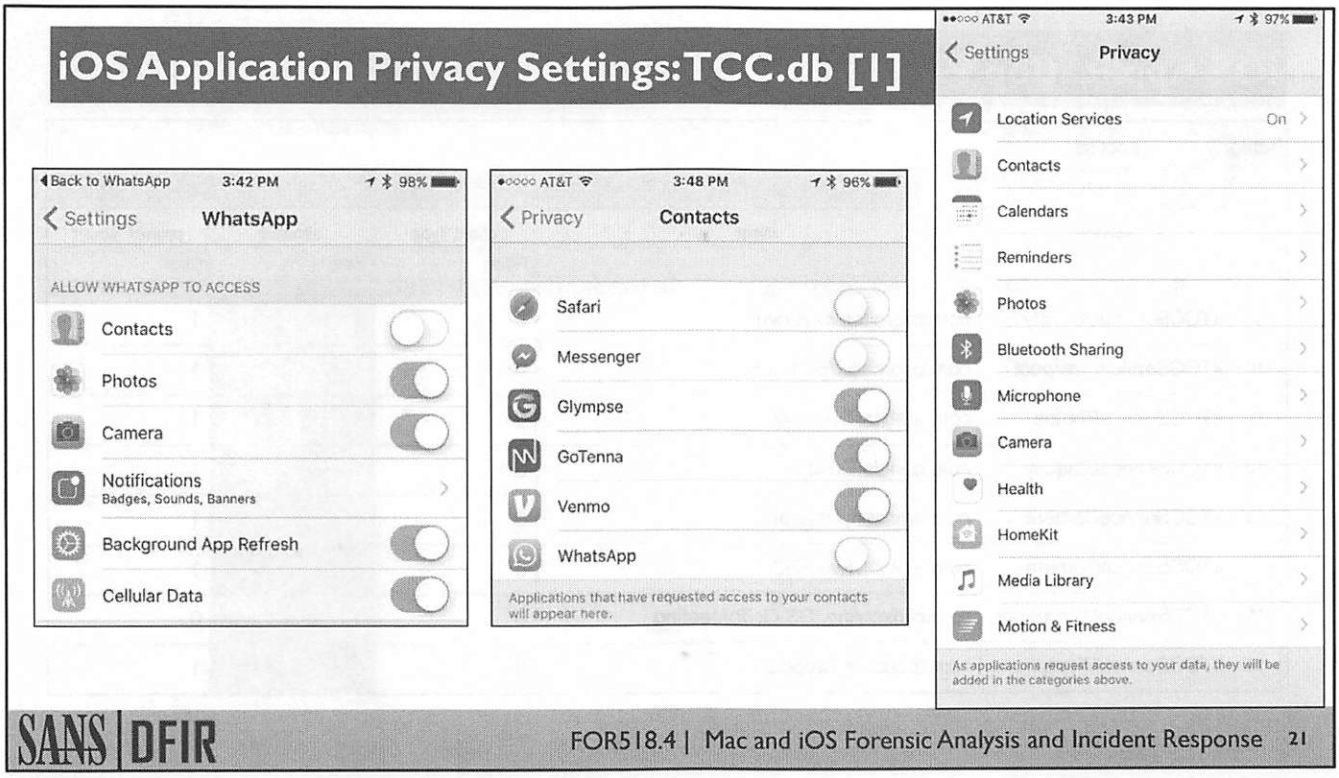


Applications on macOS and iOS ask users which permissions they may have for different capabilities available on the system. This database is named `TCC.db` and is a SQLite database.

On macOS, they are recorded in a couple of different databases: One for the system, and one for each user.

On iOS, they are stored in one database and are available in a backup or physical acquisition.

© 2020 Sarah Edwards



The `TCC.db` database contains application privacy settings. This database gets its name from Transparency, Consent, and Control.

This database can be viewed to determine what a specific application has access to. In the screenshot to the right, an application may have access to some of the following items: Location, Contacts, Calendars, Reminders, Photos, Bluetooth, Microphone, Camera, Health/Fitness, Media, HomeKit, etc.

Each application's settings can be viewed in that particular configuration in the Settings area on the device, as shown in the screenshot to the left. The WhatsApp application on this device has access to Photos and Camera, but it does not have access to Contacts.

A certain type of permission can also be viewed to determine which applications have access. The middle screenshot shows an example of this. The Contacts database is only available to the Glypmse, GoTenna, and Venmo applications but not Safari, Facebook Messenger, or WhatsApp.

References:
`https://developer.apple.com/videos/play/wwdc2016/705/`
`https://developer.apple.com/videos/play/wwdc2016/709/`

© 2020 Sarah Edwards

iOS Application Privacy Settings: TCC.db [2]

Table: access

	service	client	client_type	allowed	prompt_count
	Filter	Filter	Filter	Filter	Filter
9	kTCCServiceUbiquity	com.microsoft.skydrive	0	1	1
10	kTCCServiceLiverpool	com.ubercab.UberClient	0	1	1
11	kTCCServicePhotos	com.atebits.Tweetie2	0	1	1
12	kTCCServiceUbiquity	com.google.Drive	0	1	1
13	kTCCServiceCamera	com.amazon.Amazon	0	1	1
14	kTCCServiceCamera	com.atebits.Tweetie2	0	0	1
15	kTCCServiceMicrophone	com.citrixonline.iOS.GoToMeeting	0	1	0
16	kTCCServiceMicrophone	com.dropcam.Dropcam	0	0	1

The `TCC.db` SQLite database file keeps track of what app has which permissions or "services" as it is labeled in the database.

The screenshot above shows the following:

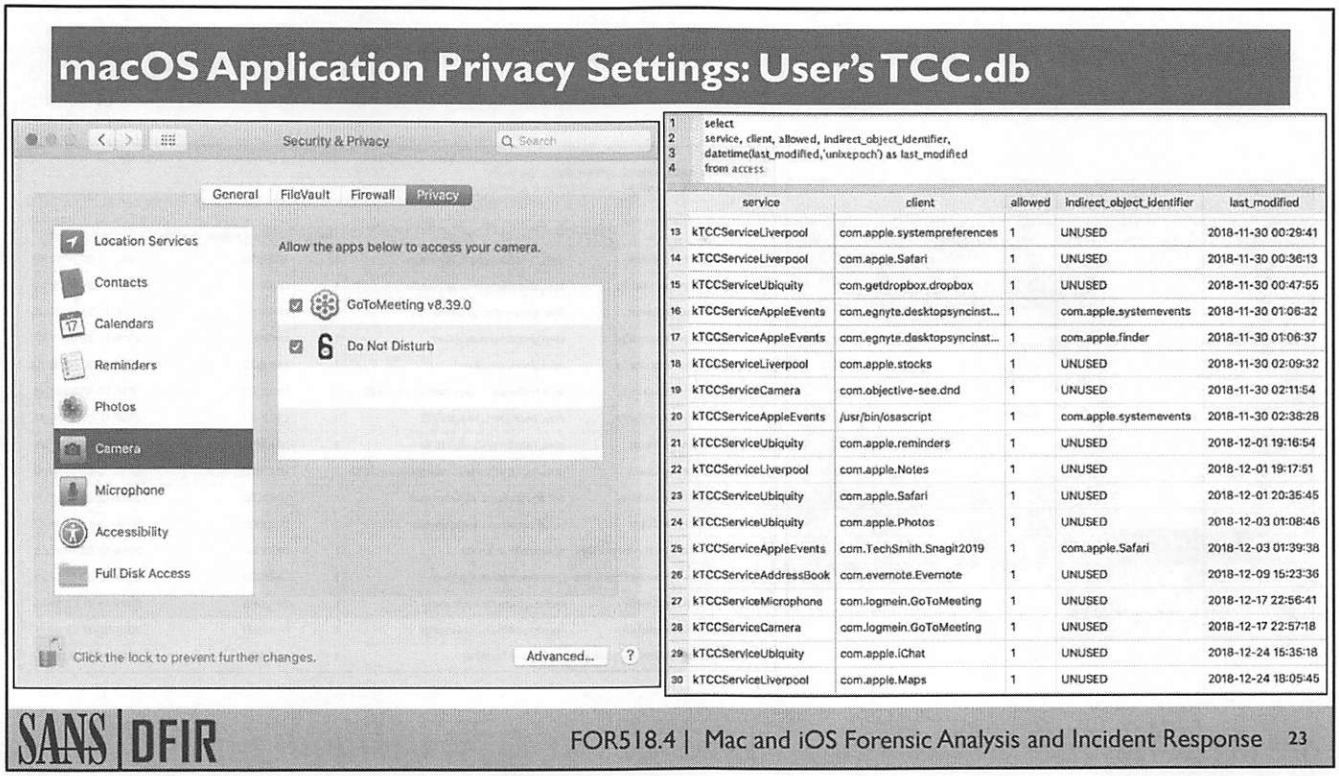
- Microsoft SkyDrive (`com.microsoft.skydrive`) has access to the iCloud "Ubiquity" Service
- Uber (`com.ubercab.UberClient`) has access to the "Liverpool" service
- Twitter (`com.atebits.Tweetie2`) has access to Photos, but no access to the Camera
- Google Drive (`com.google.Drive`) has access to iCloud
- Amazon (`com.amazon.Amazon`) has access to the Camera, probably to scan product codes
- GoToMeeting (`com.citrixonline.iOS.GoToMeeting`) has access to the microphone
- Nest DropCam (`com.dropcam.Dropcam`) DOES NOT have access to the microphone

Services may include the following:

- `kTCCServiceAll`
- `kTCCServiceAccessibility`
- `kTCCServiceCalendar`
- `kTCCServiceAddressBook`
- `kTCCServiceLocation`
- `kTCCServiceReminders`
- `kTCCServiceFacebook`
- `kTCCServiceLinkedIn`
- `kTCCServiceTwitter`
- `kTCCServiceSinaWeibo`
- `kTCCServiceLiverpool`
- `kTCCServiceUbiquity`
- `kTCCServiceTencentWeibo`

References:
`https://developer.apple.com/videos/play/wwdc2016/705/`
`https://developer.apple.com/videos/play/wwdc2016/709/`

© 2020 Sarah Edwards



The macOS privacy settings found in `TCC.db` are taken from `System Preferences | Security & Privacy | Privacy`.

For this macOS user example, two applications are using the Camera permission, GoToMeeting and Objective See's Do Not Disturb. In 10.14, the last modification time was added.

Other privacy settings may use the Address Book, Calendar, Reminders, Diagnostics, and/or various social media credentials such as Facebook and Twitter.

© 2020 Sarah Edwards

macOS Application Privacy Settings: System TCC.db


```
1  select
2  service, client, allowed, indirect_object_identifier,
3  datetime(last_modified,'unixepoch') as last_modified
4  from access
```

	service	client	allowed	indirect_object_identifier	last_modified
1	kTCCServicePostEvent	com.divisiblebyzero.Spectacle	1	UNUSED	2018-11-30 01:19:59
2	kTCCServiceAccessibility	com.divisiblebyzero.Spectacle	1	UNUSED	2018-11-30 01:19:59
3	kTCCServicePostEvent	com.getdropbox.dropbox	1	UNUSED	2018-11-30 01:26:55
4	kTCCServiceAccessibility	com.getdropbox.dropbox	1	UNUSED	2018-11-30 02:11:10
5	kTCCServiceAccessibility	com.vmware.fusion	1	UNUSED	2018-11-30 02:38:48
6	kTCCServiceAccessibility	com.techsmith.snagit.capturehelper2019	1	UNUSED	2018-12-01 21:26:41
7	kTCCServiceAccessibility	com.TechSmith.Snagit2019	1	UNUSED	2018-12-01 21:26:41
8	kTCCServicePostEvent	com.TechSmith.Snagit2019	1	UNUSED	2018-12-01 21:27:09
9	kTCCServicePostEvent	com.techsmith.snagit.capturehelper2019	1	UNUSED	2018-12-03 01:37:13
10	kTCCServiceAccessibility	net.sourceforge.sqlitebrowser	1	UNUSED	2018-12-03 03:03:11
11	kTCCServicePostEvent	net.sourceforge.sqlitebrowser	1	UNUSED	2018-12-03 03:51:08
12	kTCCServiceSystemPolicyAllFiles	com.apple.dt.Xcode	1	UNUSED	2018-12-09 15:17:13
13	kTCCServiceSystemPolicyAllFiles	com.macroplant.iExplorer	1	UNUSED	2018-12-09 15:17:21
14	kTCCServicePostEvent	com.logmein.GoToMeeting	1	UNUSED	2018-12-17 22:57:11
15	kTCCServiceAccessibility	com.logmein.GoToMeeting	1	UNUSED	2018-12-17 22:57:11
16	kTCCServiceSystemPolicyAllFiles	com.apple.Terminal	1	UNUSED	2018-12-30 03:06:01

SANS | DFIR FOR518.4 | Mac and iOS Forensic Analysis and Incident Response 24

The macOS privacy settings found in `TCC.db` are taken from `System Preferences | Security & Privacy | Privacy`.

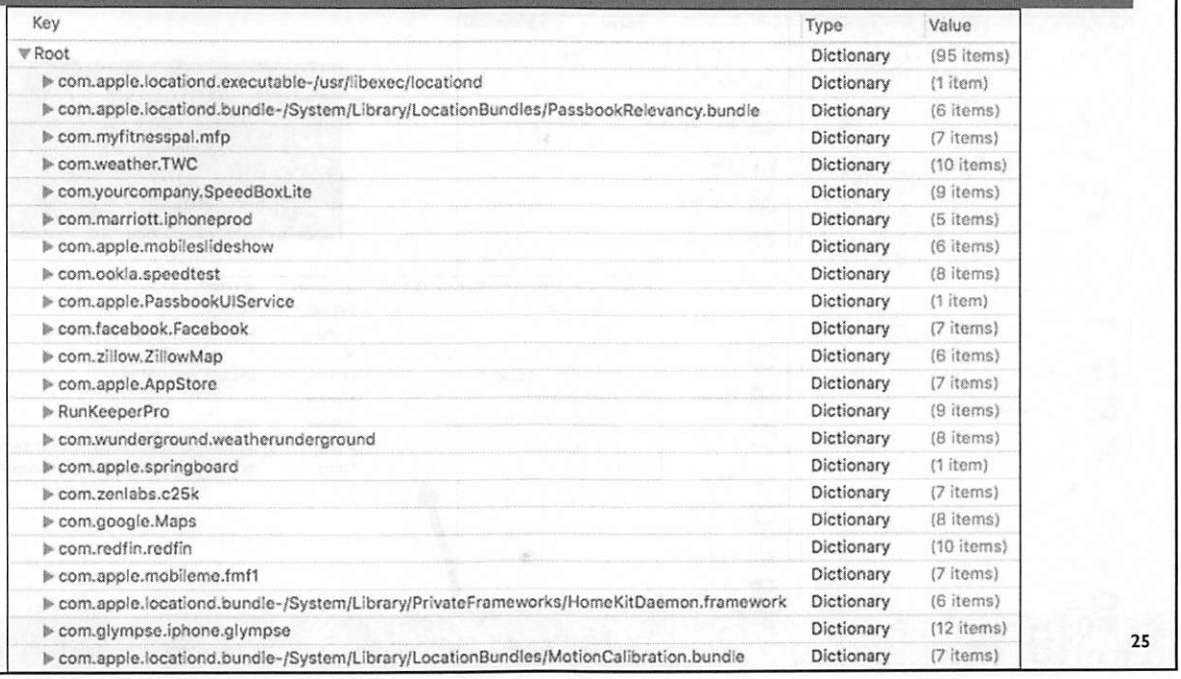
For this macOS System example, 10.14 added more permissions-based configurations. The one forensic analysts may run into the most often is the 'Full Disk Access' permission. This will limit access to some file system directories and files unless the application is provided access. In the example above, three apps have been given access:

- Terminal.app
- Xcode.app
- iExplorer.app

The database shows this access with the service "`KTCCServiceSystemPolicyAllFiles`".

© 2020 Sarah Edwards

iOS Applications Using Location Services: clients.plist



Key	Type	Value
▼ Root	Dictionary	(95 items)
▶ com.apple.locationd.executable-/usr/libexec/locationd	Dictionary	(1 item)
▶ com.apple.locationd.bundle-/System/Library/LocationBundles/PassbookRelevancy.bundle	Dictionary	(6 items)
▶ com.myfitnesspal.mfp	Dictionary	(7 items)
▶ com.weather.TWC	Dictionary	(10 items)
▶ com.yourcompany.SpeedBoxLite	Dictionary	(9 items)
▶ com.marriott.iphoneprod	Dictionary	(5 items)
▶ com.apple.mobileslideshow	Dictionary	(6 items)
▶ com.ookla.speedtest	Dictionary	(8 items)
▶ com.apple.PassbookUIService	Dictionary	(1 item)
▶ com.facebook.Facebook	Dictionary	(7 items)
▶ com.zillow.ZillowMap	Dictionary	(6 items)
▶ com.apple.AppStore	Dictionary	(7 items)
▶ RunKeeperPro	Dictionary	(9 items)
▶ com.wunderground.weatherunderground	Dictionary	(8 items)
▶ com.apple.springboard	Dictionary	(1 item)
▶ com.zenlabs.c25k	Dictionary	(7 items)
▶ com.google.Maps	Dictionary	(8 items)
▶ com.redfin.redfin	Dictionary	(10 items)
▶ com.apple.mobileme.fmf1	Dictionary	(7 items)
▶ com.apple.locationd.bundle-/System/Library/PrivateFrameworks/HomeKitDaemon.framework	Dictionary	(6 items)
▶ com.glympse.iphone.glympse	Dictionary	(12 items)
▶ com.apple.locationd.bundle-/System/Library/LocationBundles/MotionCalibration.bundle	Dictionary	(7 items)

25

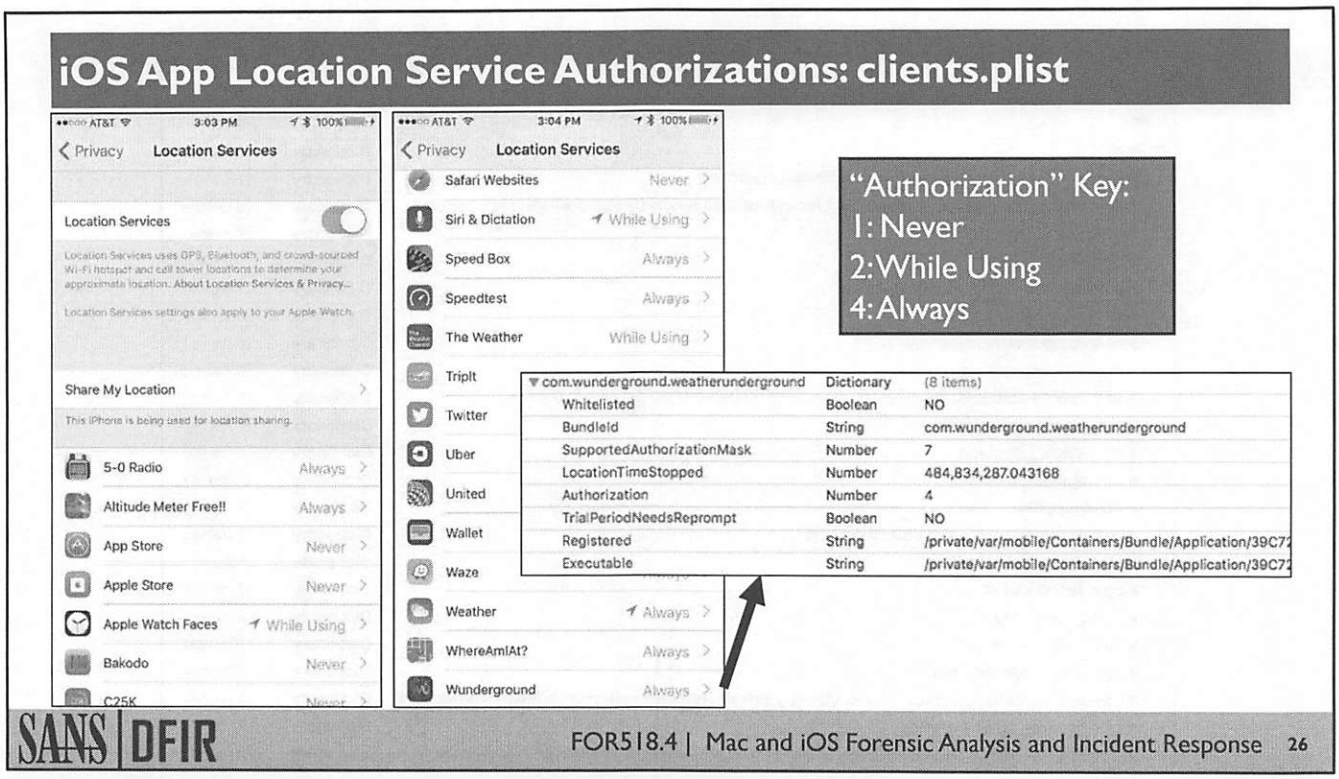
Backup: `/root/Library/Caches/locationd/clients.plist`
Physical: `/private/var/root/Library/Caches/locationd/clients.plist`

The locationd directory contains several files related to location information stored on an iOS device. The most notable may be the `consolidated.db` file, which was a cause for major concern when its contents were made public to many unsuspecting iOS users. This file contained wireless hotspots and cell tower locations within a certain radius of the device, which was said to better allow the device to calculate its location rather than relying solely on GPS satellite data. This file once contained up to a year's worth of location data, but has since been revised in subsequent iOS firmware releases.

The most valuable file in the locationd directory now is the `clients.plist` file. This file maintains a list of all of the applications that have been granted GPS permissions. This is important because it may clue the examiner to pay extra detail to those applications listed in the file, as GPS coordinates are most likely stored along with other relevant user information.

Reference:
`https://www.apple.com/newsroom/2011/04/27Apple-Q-A-on-Location-Data/`

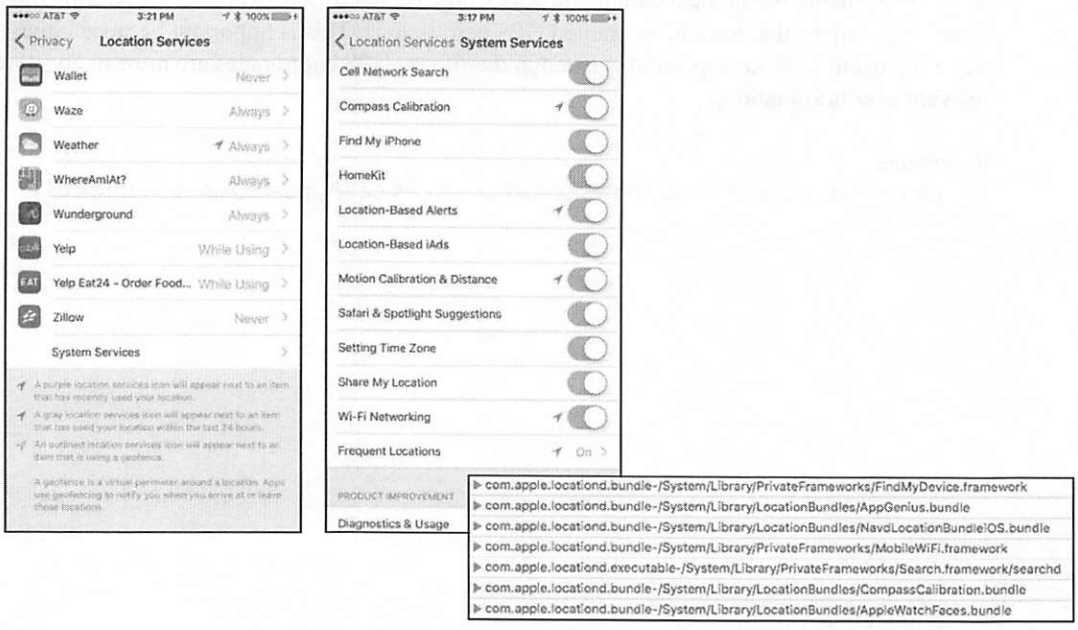
© 2020 Sarah Edwards



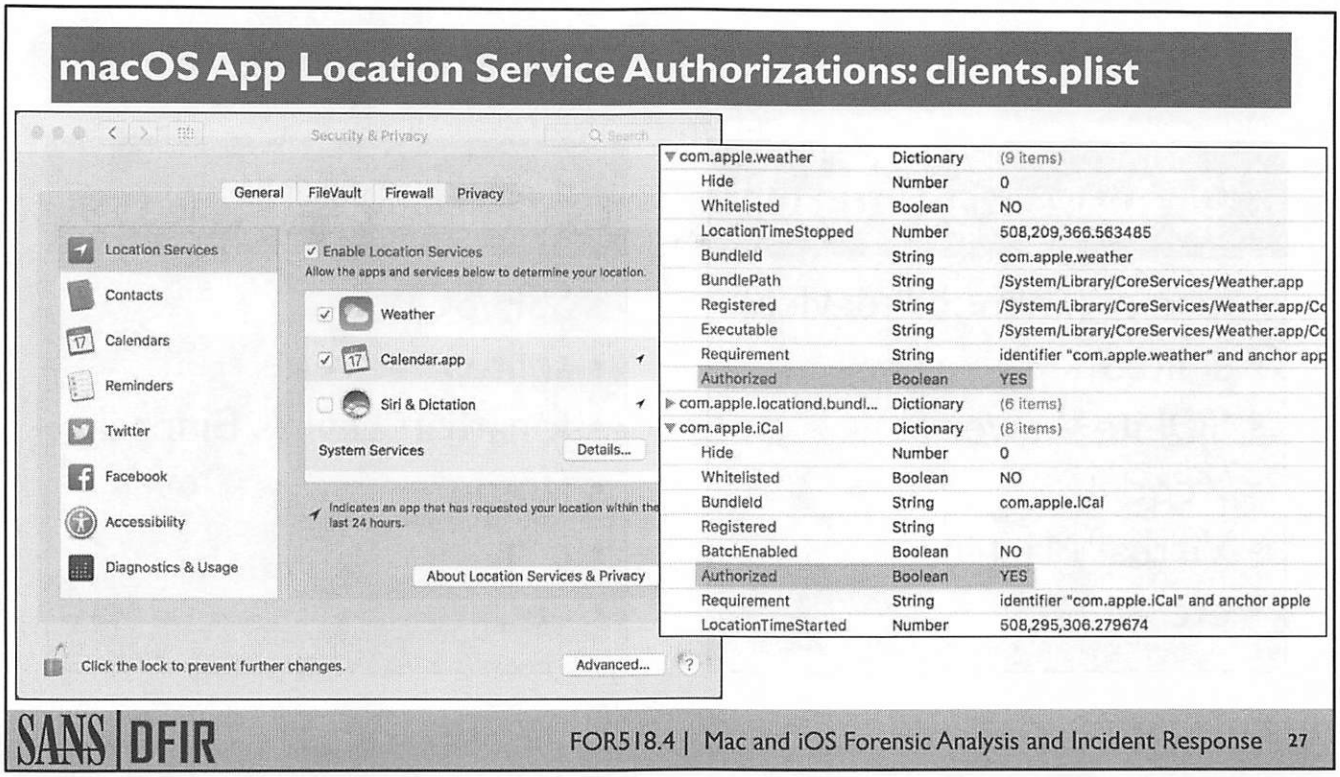
Each application's location services information is detailed in the `clients.plist` file. This plist file contains the "`LocationTimeStopped`" key, which keeps track of the time in Mac Epoch of when the application last used Location Services to get the location of the device. The key "`Authorization`" shows the level of permission for location services for that application.

1: Never authorized (no Location Services)
2: Only use Location Services when using the application
4: Always use Location Services (even when the application is backgrounded)

Along with applications, OS services can also use Location Services to notify the user of traffic conditions or to find their lost iPhone.

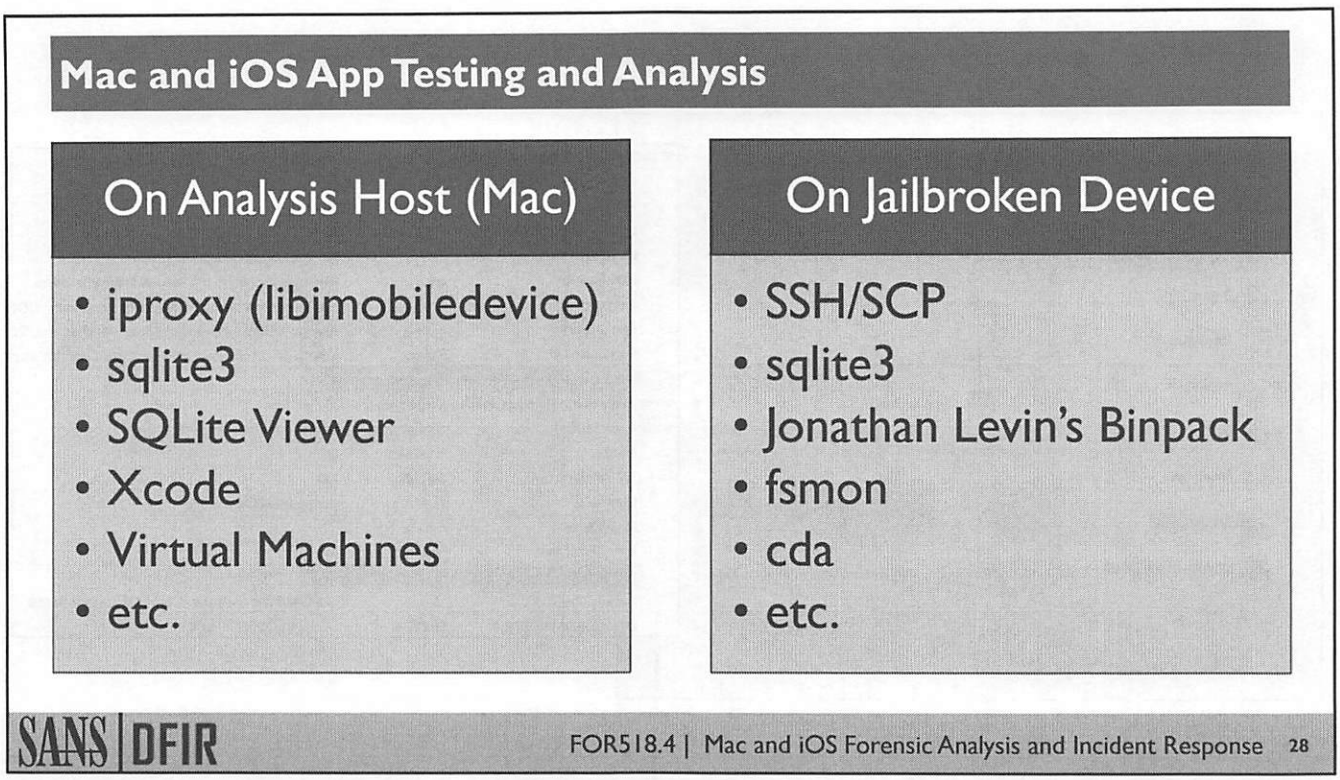


© 2020 Sarah Edwards



Just like with iOS, the location information is stored in the `clients.plist` file on macOS. This file is located in the `/private/var/db/locationd/` directory.

© 2020 Sarah Edwards



Third-party (or native!) app analysis can be done in a variety of ways, using both command-line tools as well as GUI-based tools.

On the host analysis system (for this example, we will use a Mac) we can install the `libimobiledevice` (`http://www.libimobiledevice.org/`) suite of utilities. The easiest way to install these tools is to install Homebrew first (`http://brew.sh/`), then use the command "`brew install libimobiledevice`". This will install lots of tools, including `iproxy` and `ifuse`. `iproxy` can be used to connect via USB to an analysis device (jailbroken w/SSH installed). Then SSH can be used to connect to the device for analysis without the need to connect over Wi-Fi.

```
iproxy 4242 22
ssh root@127.0.0.1 -p 4242
```

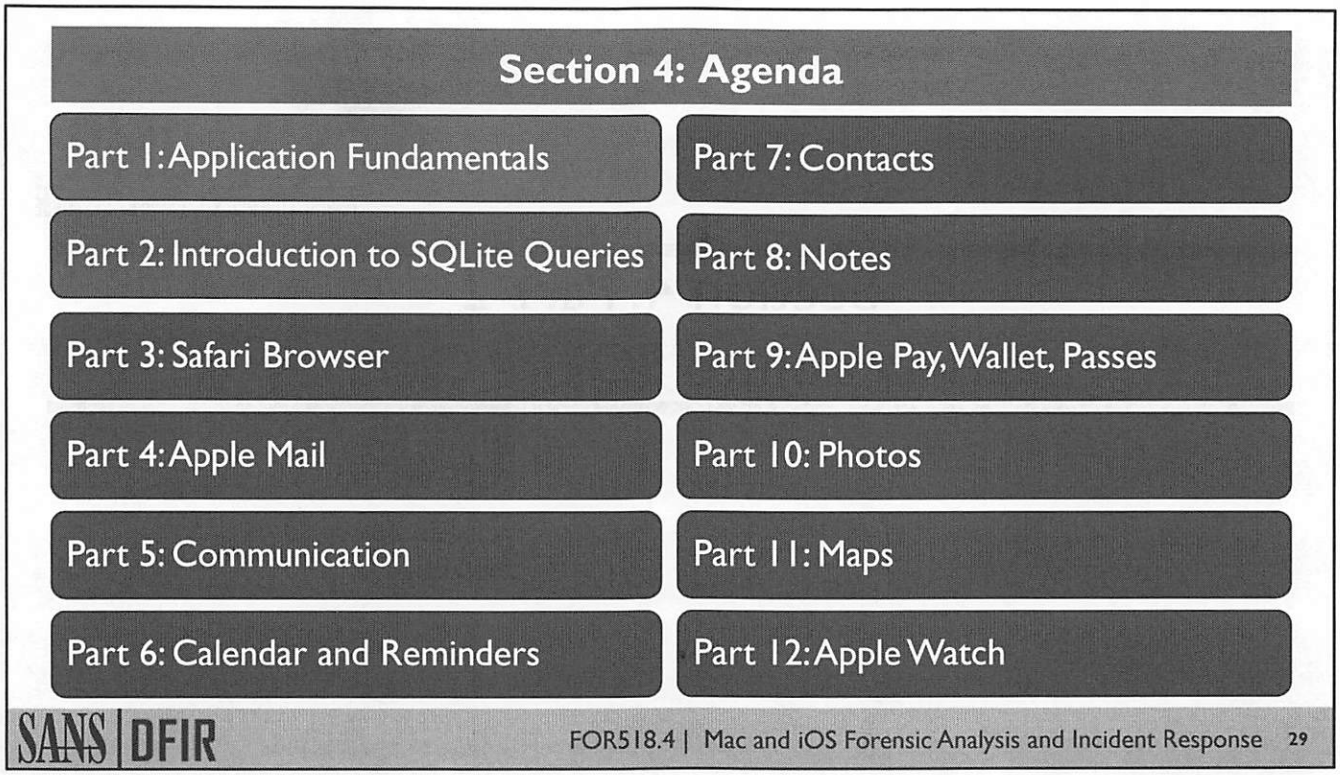
Other utilities on the host will include viewing applications like SQLite (GUI or CLI), and Xcode for plist files.

On the iDevice itself, you will want to use a jailbroken device to get full root access to the system. You never know where files are going to be stored! In the test device, you will want to at least install SSH server (some jailbreaks come with it) to get easy access to the device. Once you install SSH, make sure you change the default password from "`alpine`" as soon as possible, to protect from others getting on your test device.

Other utilities include basic UNIX commands contained in Jonathan Levin's binpack (`newosxbook.com`), fsmon for file system monitoring (`https://github.com/nowsecure/fsmon`), and CDA (`https://github.com/ay-kay/cda`) to find where app data is stored.

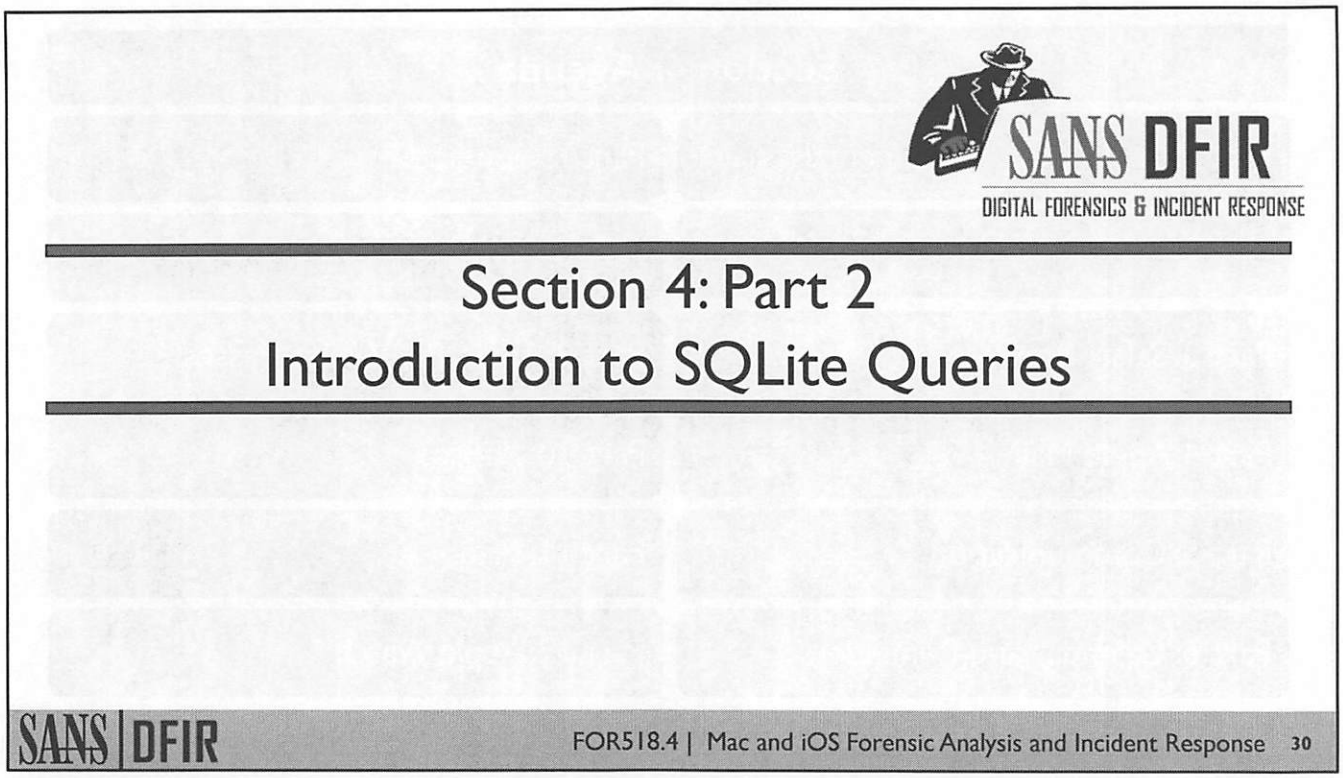
Reference:
`https://www.mac4n6.com/blog/2018/11/25/do-it-live-dynamic-ios-forensic-testing`

© 2020 Sarah Edwards



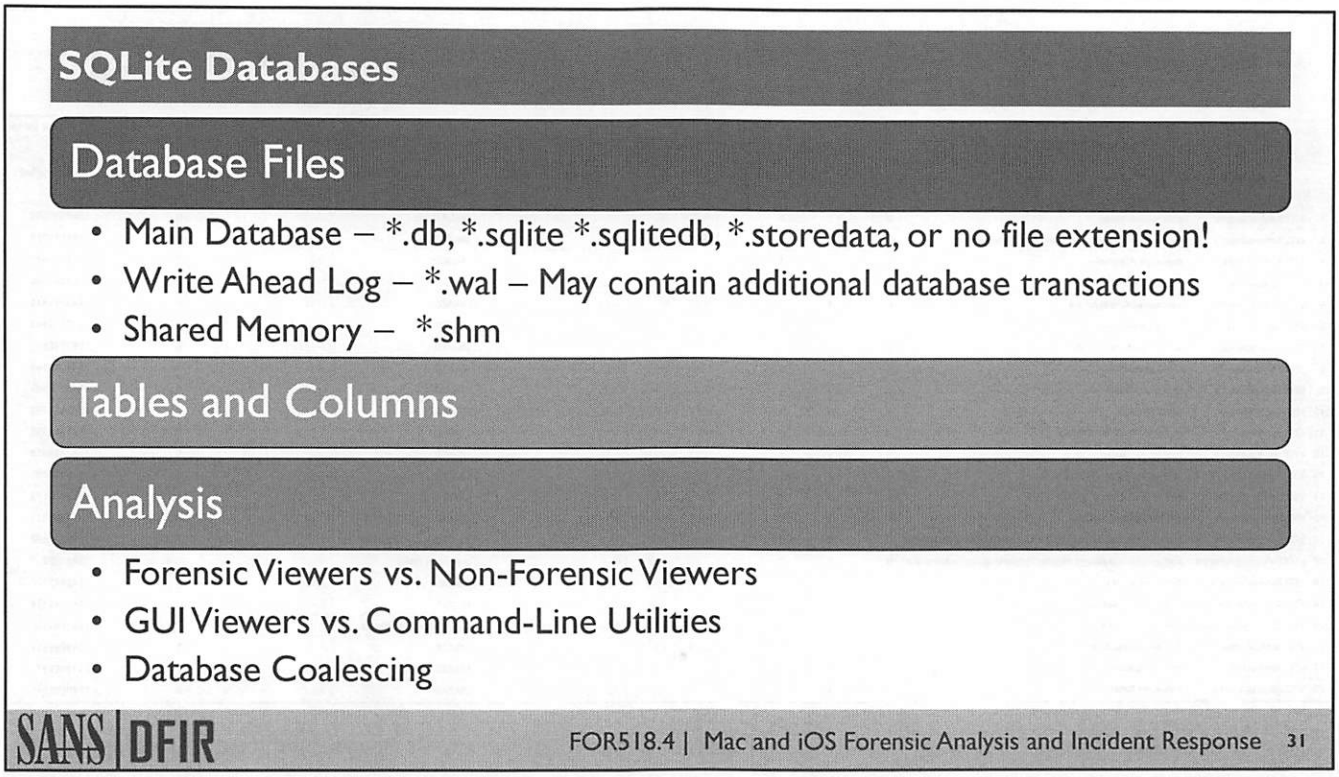
This page intentionally left blank.

© 2020 Sarah Edwards



This page intentionally left blank.

© 2020 Sarah Edwards



SQLite databases are made up of multiple files. The main database may have a variety of file extensions or none at all. Looking for that "`SQLite format 3`" header can help you determine if a file is a SQLite database. The Write Ahead Log or WAL file is important as it may contain additional database transactions not yet coalesced into the main database file. The Shared Memory file (`*.shm`) likely does not contain any database transaction but is used to help facilitate transactions into the WAL file.

As with other databases, SQLite databases can be comprised from multiple tables, each table containing different data in columns. Some databases may have one table, others may have hundreds. Most database schemas are quite different from each other. The data in the columns will be a variety of different datatypes.

There are many types of SQLite viewers out there to use for analysis. Some are GUI-based, others can be run from the command line. Most are not forensic-based viewers and may coalesce the transactions in the WAL file into the main database, like DB Browser for SQLite. This is fine as long as the analyst is aware that is happening or not. Forensic-based utilities may allow the analyst to choose to coalesce or not. Forensic-based browsers may also attempt to recover database entries.

References:
`https://www.sqlite.org/docs.html`
`http://www.sqlitetutorial.net`

© 2020 Sarah Edwards

Example Database – TCC.db

Table: access

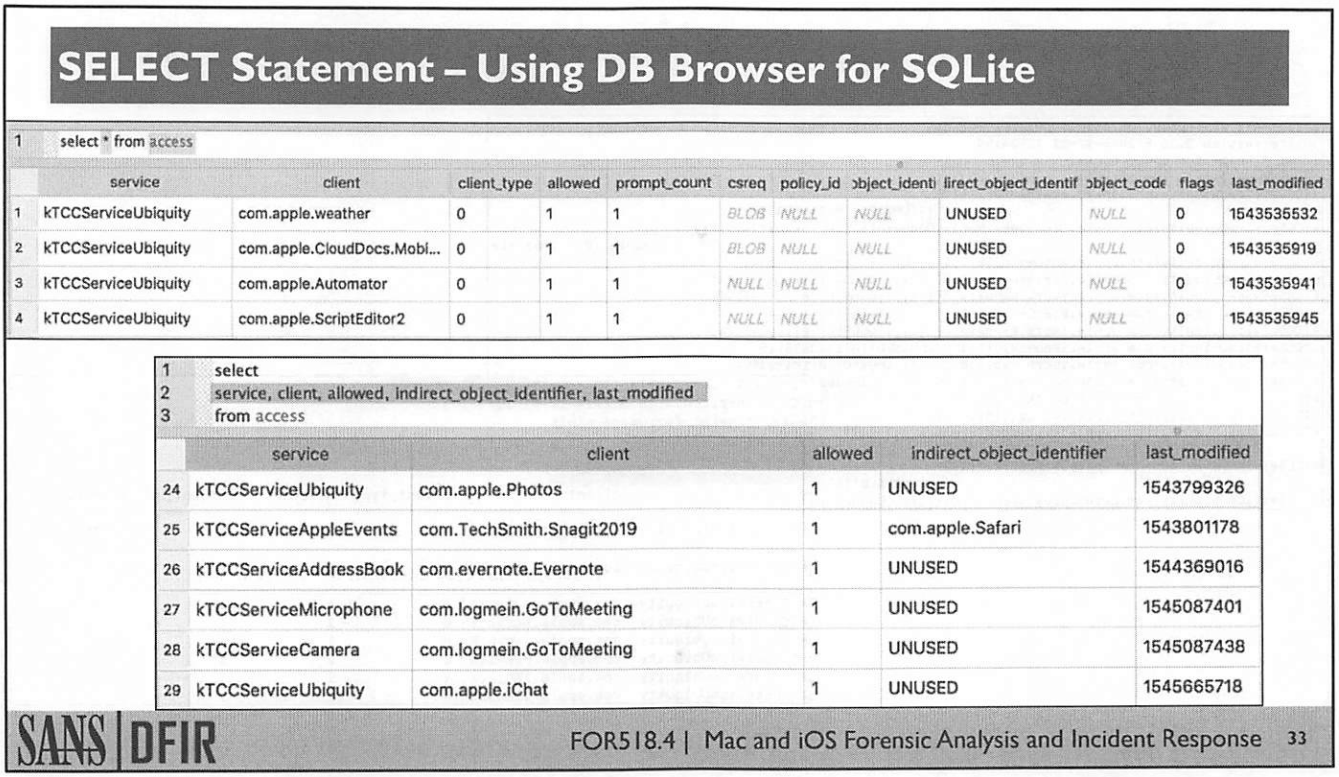
New Recor

	service	client	client_type	allowed	prompt_count	csreq	policy_id	indirect_object_identifier_type	indirect_object_identifier	indirect_object_code_identity	flags	last_modified
	Filter	Filter	Filter	Fil...	Filter	...	Fil...	Filter	Filter	Filter	Filter	Filter
1	kTCCServiceUbiquity	com.apple.weather	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	0	1543535532
2	kTCCServiceUbiquity	com.apple.CloudDocs.MobileDocumentsFileProvider	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	0	1543535919
3	kTCCServiceUbiquity	com.apple.Automator	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535941
4	kTCCServiceUbiquity	com.apple.ScriptEditor2	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535945
5	kTCCServiceUbiquity	com.apple.iWork.Numbers	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535945
6	kTCCServiceUbiquity	com.apple.Preview	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535945
7	kTCCServiceUbiquity	com.apple.iWork.Keynote	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535945
8	kTCCServiceUbiquity	com.apple.iBooksX	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535945
9	kTCCServiceUbiquity	com.apple.iWork.Pages	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535945
10	kTCCServiceUbiquity	com.apple.mail	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535953
11	kTCCServiceUbiquity	com.apple.QuickTimePlayerX	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535955
12	kTCCServiceUbiquity	com.apple.TextEdit	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543535958
13	kTCCServiceLiverpool	com.apple.systempreferences	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	0	1543537781
14	kTCCServiceLiverpool	com.apple.Safari	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	0	1543538173
15	kTCCServiceUbiquity	com.getdropbox.dropbox	0	1	1	NULL	NULL	NULL	UNUSED	NULL	0	1543538875
16	kTCCServiceAppleEvents	com.egnyte.desktopsyncinstaller.Egnyte-Desktop-Sync-Installer	0	1	1	BLOB	NULL	0	com.apple.systemevents	BLOB	NULL	1543539992
17	kTCCServiceAppleEvents	com.egnyte.desktopsyncinstaller.Egnyte-Desktop-Sync-Installer	0	1	1	BLOB	NULL	0	com.apple.finder	BLOB	NULL	1543539997
18	kTCCServiceLiverpool	com.apple.stocks	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	0	1543543772
19	kTCCServiceCamera	com.objective-see.dnd	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	NULL	1543543914
20	kTCCServiceAppleEvents	/usr/bin/osascript	1	1	1	BLOB	NULL	0	com.apple.systemevents	BLOB	NULL	1543545508
21	kTCCServiceUbiquity	com.apple.reminders	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	0	1543691814
22	kTCCServiceLiverpool	com.apple.Notes	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	0	1543691871
23	kTCCServiceUbiquity	com.apple.Safari	0	1	1	BLOB	NULL	NULL	UNUSED	NULL	0	1543696545

SANS | DFIR FOR518.4 | Mac and iOS Forensic Analysis and Incident Response 32

The database above is an example that will be used for most of the following query examples. This `TCC.db` database is a good starter database to use because it is relatively simple. We will be looking at one table, '`access`' to review the permissions for various applications on a macOS system.

© 2020 Sarah Edwards



The `SELECT` SQLite statement is the basis of most SQL queries that will extract data from a database. `SELECT` is used to "select" different items from tables and columns.

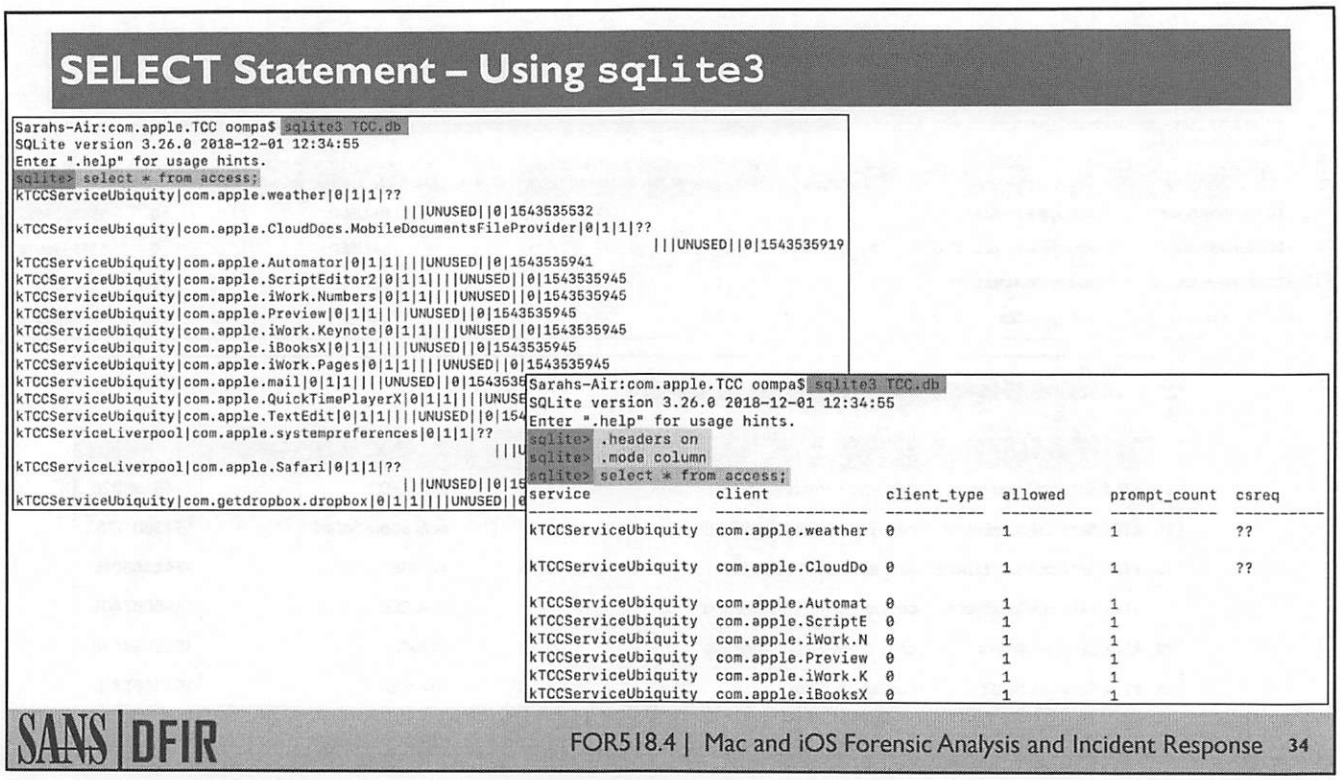
In the top example, a wildcard (`*`) is used to "select" everything from the "`access`" table. This is the most broad example, but also useful to filter down columns to what is of investigative importance while in a query building window of DB Browser for SQLite.

The second example shows data being extracted from only certain columns; `service`, `client`, `allowed`, `indirect_object_identifier`, and `last_modified`.

It is worth mentioning that once queries start becoming more complicated with table JOINs, you will want to preface the column name with a specific table to differentiate columns that may have the same name across tables (i.e.: `access.service`, `access.client`, `access.allowed` and so on).

Reference:
`https://www.sqlite.org/docs.html`

© 2020 Sarah Edwards



If you want to use `sqlite3` on the command line to do your analysis, that is another option. Performing queries on the command line can be done in a couple of ways.

The examples show a query being performed in the SQLite shell. This is an interactive shell specifically for SQLite databases. The default output in the top example is not exactly the easiest to visually parse. Using a few SQLite shell commands in the second example we can make the output more appealing for analysis.

- `.headers on` – This puts the column names at the top of the output.
- `.mode column` – Puts the data in columns. The column width can be configured; this example shows a default width. The data in the columns may be cut off.

Reference:
`https://www.sqlite.org/docs.html`

© 2020 Sarah Edwards

SELECT Statement – Using sqlite3, CLI One-Liner

```
Sarahs-Air:com.apple.TCC oompa$ sqlite3 TCC.db 'select * from access;'
kTCCServiceUbiquity|com.apple.weather|0|1|1|??
                                          |||UNUSED||0|1543535532
kTCCServiceUbiquity|com.apple.CloudDocs.MobileDocumentsFileProvider|0|1|1|??
                                                                  |||UNUSED||0|1543535919
kTCCServiceUbiquity|com.apple.Automator|0|1|1||||UNUSED||0|1543535941
kTCCServiceUbiquity|com.apple.ScriptEditor2|0|1|1||||UNUSED||0|1543535945
kTCCServiceUbiquity|com.apple.iWork.Numbers|0|1|1||||UNUSED||0|1543535945
```

```
Sarahs-Air:com.apple.TCC oompa$ sqlite3 -header -column TCC.db 'select * from access;'
service              client             client_type  allowed     prompt_count  csreq
-------------------  -----------------  -----------  ----------  ------------  -------
kTCCServiceUbiquity  com.apple.weather  0            1           1             ??

kTCCServiceUbiquity  com.apple.CloudDo  0            1           1             ??

kTCCServiceUbiquity  com.apple.Automat  0            1           1
kTCCServiceUbiquity  com.apple.ScriptE  0            1           1
kTCCServiceUbiquity  com.apple.iWork.N  0            1           1
kTCCServiceUbiquity  com.apple.Preview  0            1           1
```


The `sqlite3` utilities can also be used from the command line in a one-liner script (or in a more complex bash script!). The query is provided at the end of the one-liner in quotes.

Like the previous example, the output is not pretty. We can use `-header` and `-column` to fix this.

Reference:
`https://www.sqlite.org/docs.html`

© 2020 Sarah Edwards

Timestamp Conversion and Column Renaming

```
1  select
2  service, client, allowed, indirect_object_identifier,
3  datetime(last_modified,'unixepoch','localtime')
4  from access
```

	service	client	allowed	indirect_object_identifier	datetime(last_modified,'unixepoch','localtime')
24	kTCCServiceUbiquity	com.apple.Photos	1	UNUSED	2018-12-02 20:08:46
25	kTCCServiceAppleEvents	com.TechSmith.Snagit2019	1	com.apple.Safari	2018-12-02 20:39:38
26	kTCCServiceAddressBook	com.evernote.Evernote	1	UNUSED	2018-12-09 10:23:36
27	kTCCServiceMicrophone	com.logmein.GoToMeeting	1	UNUSED	2018-12-17 17:56:41
28	kTCCServiceCamera	com.logmein.GoToMeeting	1	UNUSED	2018-12-17 17:57:18

```
1  select
2  service, client, allowed, indirect_object_identifier,
3  datetime(last_modified,'unixepoch','localtime') as "Last Modified"
4  from access
```

	service	client	allowed	indirect_object_identifier	Last Modified
24	kTCCServiceUbiquity	com.apple.Photos	1	UNUSED	2018-12-02 20:08:46
25	kTCCServiceAppleEvents	com.TechSmith.Snagit2019	1	com.apple.Safari	2018-12-02 20:39:38

Adding to our initial `SELECT` query, we can manipulate some of the data to make it easier to read. One data type that you will often want to convert is epoch timestamps. Our example stores a Last Modified timestamp in Unix epoch, or number of seconds from 1/1/1970 00:00:00 UTC.

SQLite can convert this by using a `DATETIME` function. The example above is converting from `'unixepoch'` to `'localtime'`. The `'localtime'` modifier will show you the timestamp in local system time. If you wanted to keep it to UTC, remove that modifier, as it will be in UTC by default.

Once converted, the column name is long and unwieldly; this can be changed by using `AS`. The example at the bottom is changing the column name to "`Last Modified`". This can be useful to put more context to otherwise strange column names. Some database designers are better at naming columns/tables than others.

References:
`https://www.sqlite.org/docs.html`
`https://www.sqlite.org/lang_datefunc.html`

© 2020 Sarah Edwards

DISTINCT Keyword and Commenting

```
1  select
2  distinct service
3  --, client, allowed, indirect_object_identifier, datetime(last_modified,'unixepoch','localtime') as "Last Modified"
4  from access
```

	service
1	kTCCServiceAddressBook
2	kTCCServiceAppleEvents
3	kTCCServiceCamera
4	kTCCServiceLiverpool
5	kTCCServiceMicrophone
6	kTCCServiceUbiquity

The `DISTINCT` keyword is useful to find unique values in a column. Using our TCC example, if I wanted to see all the different types of permissions being used in this table I would use `DISTINCT` on the `service` column. There are six distinct permissions for this example.

`DISTICT` is here used on a single column. When doing analysis, sometimes you just want to quickly see what the values are but not delete everything in your query. This example also shows me commenting out the rest of the query by using "`--`".

Reference:
`https://www.sqlite.org/docs.html`

© 2020 Sarah Edwards

CASE Expression

```
1  select
2  service, client,
3  case allowed
4      when 0 then 'Not Allowed'
5      when 1 then 'Allowed'
6  end Allowed,
7  indirect_object_identifier,
8  datetime(last_modified,'unixepoch','localtime') as "Last Modified"
9  from access
```

	service	client	Allowed	indirect_object_identifier	Last Modified
30	kTCCServiceLiverpool	com.apple.Maps	Allowed	UNUSED	2018-12-24 13:05:45
31	kTCCServiceAppleEvents	com.blackbagtech.BlackLight	Allowed	com.apple.finder	2019-01-12 19:02:39
32	kTCCServiceAddressBook	com.microsoft.onenote.mac	Allowed	UNUSED	2019-01-18 12:15:28
33	kTCCServiceAddressBook	com.microsoft.Word	Allowed	UNUSED	2019-01-18 12:16:09
34	kTCCServiceAddressBook	com.microsoft.Powerpoint	Allowed	UNUSED	2019-01-18 12:17:15
35	kTCCServiceAppleEvents	com.logitech.presenter	Allowed	com.apple.systempreferences	2019-01-28 10:16:50
36	kTCCServiceAppleEvents	com.logitech.presenter	Allowed	com.apple.Safari	2019-01-28 10:21:03
37	kTCCServiceUbiquity	com.apple.Grab	Allowed	UNUSED	2019-01-31 16:38:24
38	kTCCServiceAddressBook	com.microsoft.Excel	Not Allowed	UNUSED	2019-02-06 20:07:05
39	kTCCServiceAppleEvents	com.logitech.presenter	Allowed	com.google.Chrome	2019-02-20 09:09:02

SANS | DFIR FOR518.4 | Mac and iOS Forensic Analysis and Incident Response 38

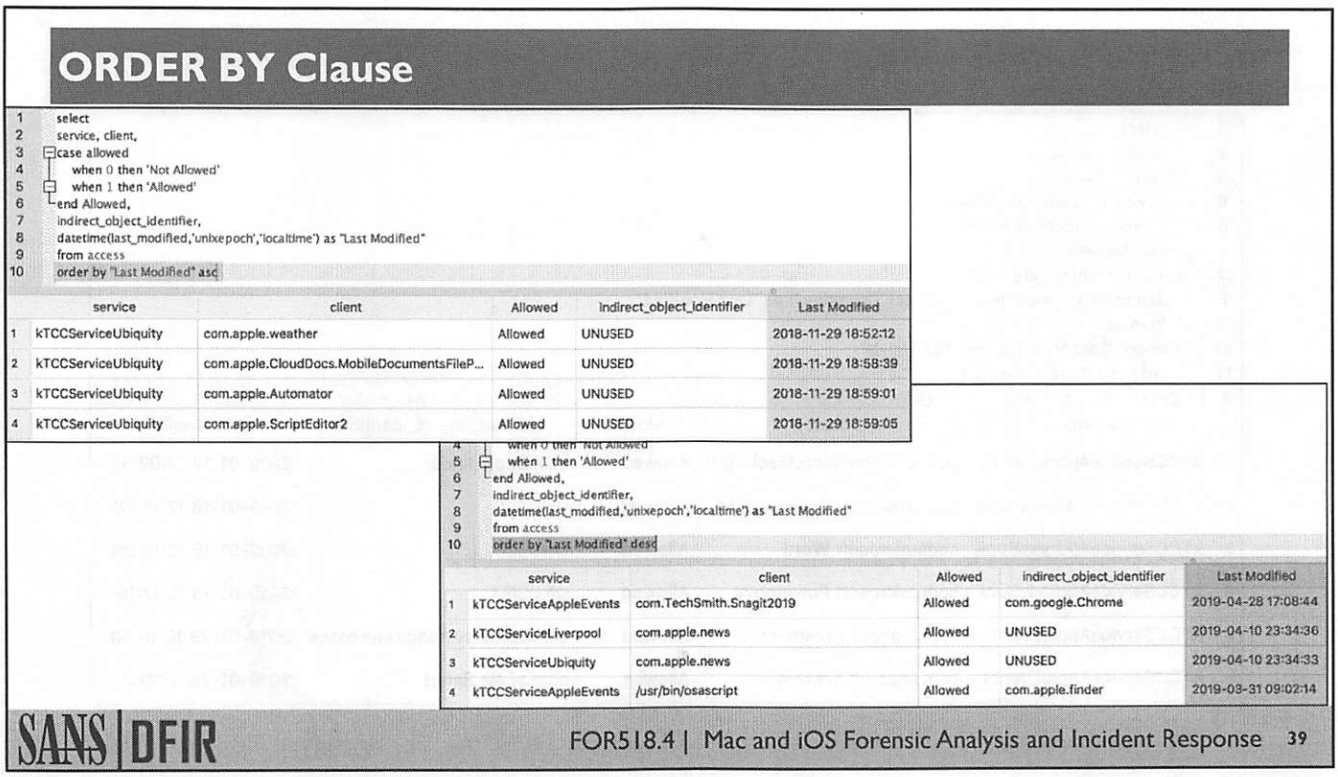
When you have certain values for different types of entries it can be useful to rename them or perform a certain action on just that value. This is where the `CASE` expression is handy.

The example above is changing a value of "`0`" in the allowed column to "`Not Allowed`" and a value of "`1`" to "`Allowed`". This can help provide context to the extracted data.

Another `CASE` example might be to change the service to something more human friendly. (i.e.: `kTCCServiceUbiquity` to "`iCloud`")

Reference:
`https://www.sqlite.org/docs.html`

© 2020 Sarah Edwards



An analysis technique could be to order certain columns, this is commonly done on timestamp columns to determine what records were performed before others.

The `ORDER BY` clause can be used to order temporally in the example above on the "`Last Modified`" column. The two examples show the same statement but different orderings.

- `ASC` – Ascending Order, this is the default if this keyword is not provided.
- `DESC` – Descending Order

Reference:
`https://www.sqlite.org/docs.html`

© 2020 Sarah Edwards

WHERE Clause and LIKE Function

```
1   select
2   service, client,
3   case allowed
4       when 0 then 'Not Allowed'
5       when 1 then 'Allowed'
6   end Allowed,
7   indirect_object_identifier,
8   datetime(last_modified,'unixepoch','localtime') as "Last Modified"
9   from access
10  where "Last Modified" like "%2019-%"
11  order by "Last Modified"
```

	service	client	Allowed	indirect_object_identifier	Last Modified
1	kTCCServiceAppleEvents	com.blackbagtech.BlackLight	Allowed	com.apple.finder	2019-01-12 19:02:39
2	kTCCServiceAddressBook	com.microsoft.onenote.mac	Allowed	UNUSED	2019-01-18 12:15:28
3	kTCCServiceAddressBook	com.microsoft.Word	Allowed	UNUSED	2019-01-18 12:16:09
4	kTCCServiceAddressBook	com.microsoft.Powerpoint	Allowed	UNUSED	2019-01-18 12:17:15
5	kTCCServiceAppleEvents	com.logitech.presenter	Allowed	com.apple.systempreferences	2019-01-28 10:16:50
6	kTCCServiceAppleEvents	com.logitech.presenter	Allowed	com.apple.Safari	2019-01-28 10:21:03

SANS | DFIR FOR518.4 | Mac and iOS Forensic Analysis and Incident Response 40

Some databases may have thousands of records and filtering is required to find specific records of interest.

Using the `WHERE` clause, we can filter on different pieces of the output. The example shows a filter for "`Last Modified`" times that start with the string "`2019-`". The '`%`' wildcards are used as part of the string matching, zero or more characters before or after the specified string.

Reference:
`https://www.sqlite.org/docs.html`

© 2020 Sarah Edwards

Table JOINs (Using Safari CloudTabs.db)

```
1  select
2  cloud_tab_devices.device_name,
3  cloud_tabs.title, cloud_tabs.url,
4  datetime(cloud_tab_devices.last_modified+978307200,'unixepoch','localtime') as last_modified
5  from cloud_tabs
6  left join cloud_tab_devices where cloud_tabs.device_uuid = cloud_tab_devices.device_uuid
```

	device_name	title	url	last_modified
1	Sarah's MacBook Air	Twitter / Notifications	https://twitter.com/i/notifications	2019-01-01 13:45:50
2	Sarah's MacBook Air	About Secure Boot - Apple Support	https://support.apple.com/en-us/HT208330	2019-01-01 13:45:50
3	Sarah's MacBook Air	Twitter	https://twitter.com/	2019-01-01 13:45:50
4	Sarah's MacBook Air	Objective-See's Blog	https://objective-see.com/blog/blog_0x3C.html	2019-01-01 13:45:50
5	Sarah's MacBook Air	Jailbreak - The iPhone Wiki	https://www.theiphonewiki.com/wiki/Jailbreak	2019-01-01 13:45:50
6	miPhoneX	Koala - Wikipedia	https://en.m.wikipedia.org/wiki/Koala	2019-01-01 13:47:51
7	miPhoneX	Mac & IOS Forensic Analysis & Incident Resp...	https://www.sans.org/course/mac-and-ios-fo...	2019-01-01 13:47:51
8	miPhoneX	london - Google Search	https://www.google.com/search?q=london&ie...	2019-01-01 13:47:51
9	miPhoneX	Apple Park Visitor Center - Apple	https://www.apple.com/retail/appleparkvisitor...	2019-01-01 13:47:51

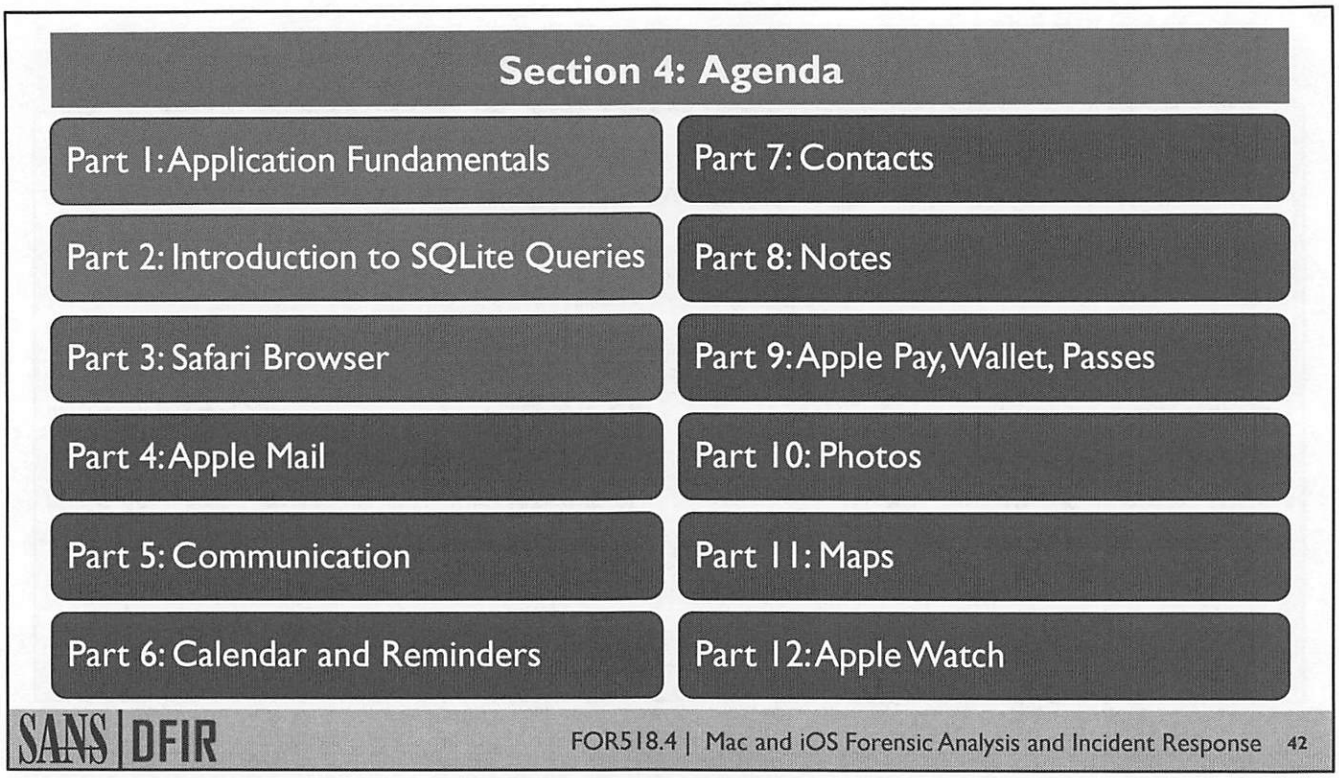
Table `JOIN`s are used to combine data between two tables. These joins are performed on a common piece of data in each table.

The example above shows the `CloudTabs.db` Safari database. The table `cloud_tabs` contains the `title` and `url` or synced Safari tabs, while the table `cloud_tab_devices` contains the `device_name`, and `last_modified` timestamp.

The table join is done with a `device_uuid` (not shown) column in each table.

Reference:
`https://www.sqlite.org/docs.html`

© 2020 Sarah Edwards



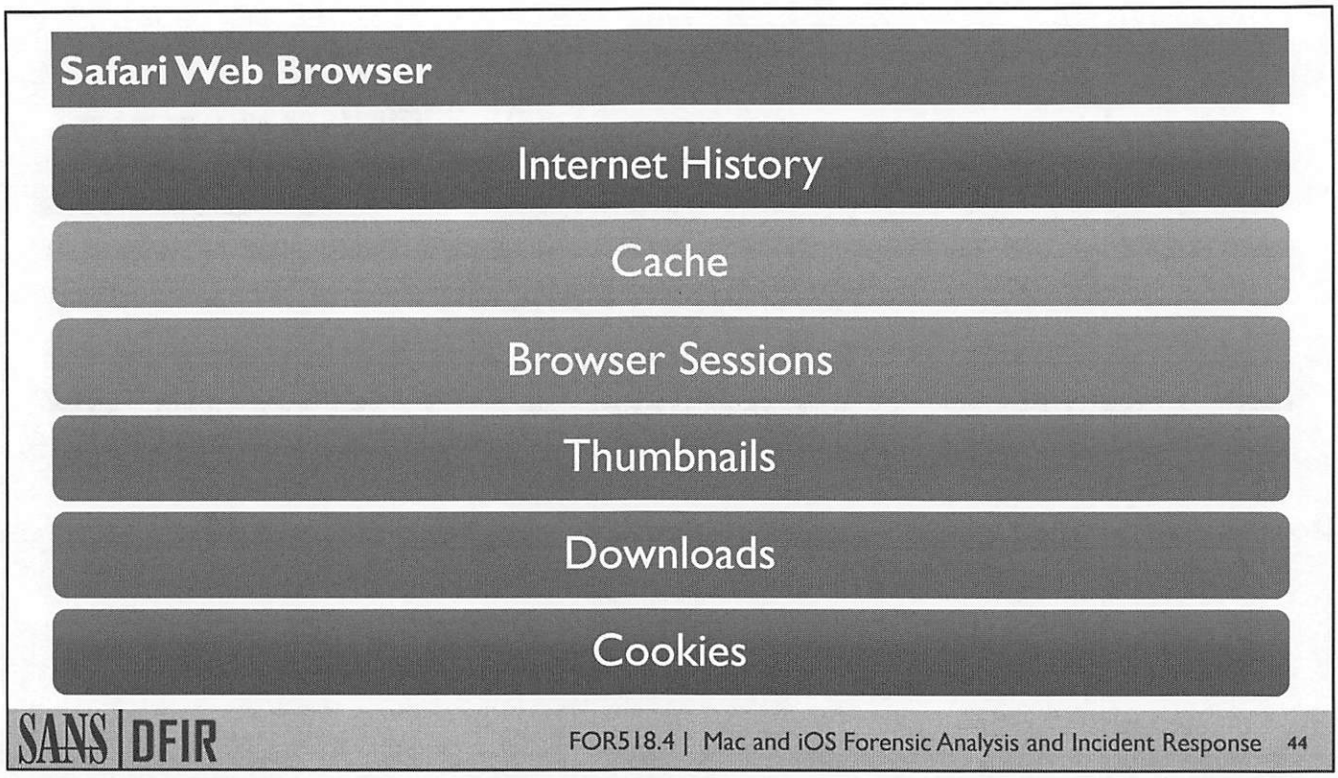
This page intentionally left blank.

© 2020 Sarah Edwards

Section 4: Part 3
Safari Browser

This page intentionally left blank.

© 2020 Sarah Edwards



iOS Physical:

- iOS 8+: `/private/var/mobile/Containers/Data/Application/<GUID>/Library/Safari/`
- iOS 7-: `/private/var/mobile/Applications/<GUID>/Library/Safari/`
- iOS 6-: `/private/var/mobile/Library/Safari/`

iOS Backup/FS:

- iOS 8+: `/mobile/Applications/com.apple.mobilesafari/Library/Safari/`
- iOS 7: `/mobile/Applications/com.apple.mobilesafari/Library/Safari/`
- iOS 6: `/mobile/Library/Safari/`

iOS Backup/FS:

- `~/Library/Safari/`
- `~/Library/Caches/com.apple.Safari/`

© 2020 Sarah Edwards

Safari data on iOS or macOS devices has changed over time; it was in a plist file and has now migrated to a SQLite database. The various directories and files will be discussed, but it is worth taking a look at the preferences files `com.apple.safarimobile.plist/com.apple.safari.plist` or `RecentSearches.plist` (iOS 7-) for Recent Web Searches (Date/Search String), the `Bookmarks.db/Bookmarks.plist` file for Bookmarks, and the binary cookies files as well.

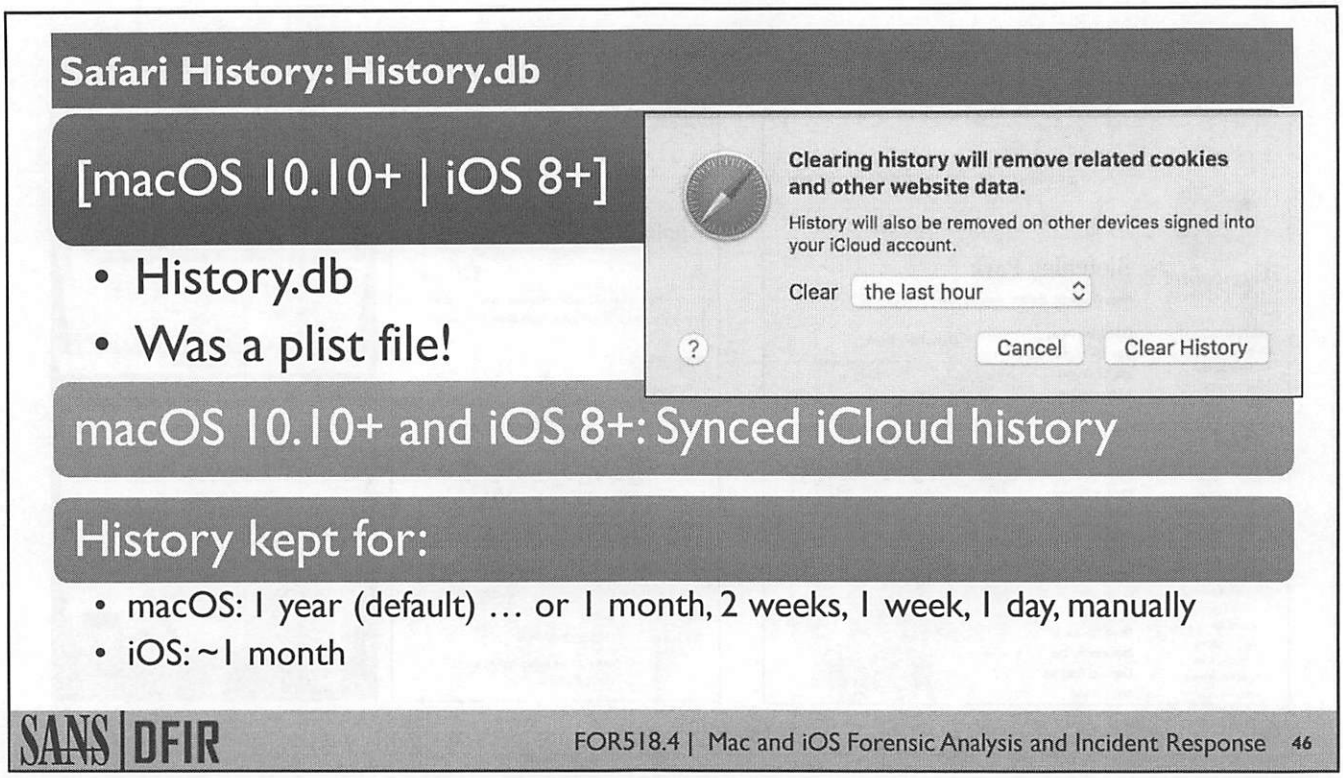
The web browser native to OS X since 10.3 (before this, the default was Internet Explorer!) is the Safari web browser, while the Mobile Safari browser has been on iOS since the beginning.

Other browsers may also be used, such as Firefox, Google Chrome, and Opera; however, we will only cover the native browser in this class, as other web browsers tend to be platform independent.

Below is an example of recent web searches that are stored in the `com.apple.Safari.plist` file in the user's macOS preferences directory.

▼RecentWebSearches	Array	(3 items)
▼Item 0	Dictionary	(2 items)
SearchString	String	penguins
Date	Date	Dec 15, 2014, 3:34:11 AM
▼Item 1	Dictionary	(2 items)
SearchString	String	mac pro
Date	Date	Dec 15, 2014, 3:33:42 AM
▼Item 2	Dictionary	(2 items)
SearchString	String	chrome
Date	Date	Dec 14, 2014, 4:50:41 PM

© 2020 Sarah Edwards



macOS:

- 10.10+: `~/Library/Safari/History.db`
- 10.9-: `~/Library/Safari/History.plist`

iOS Physical:

- iOS 8+: `/private/var/mobile/Containers/Data/Application/<GUID>/Library/Safari/History.db`
- iOS 7-: `/private/var/mobile/Applications/<GUID>/Library/Safari/History.plist`
- iOS 6-: `/private/var/mobile/Library/Safari/History.plist`

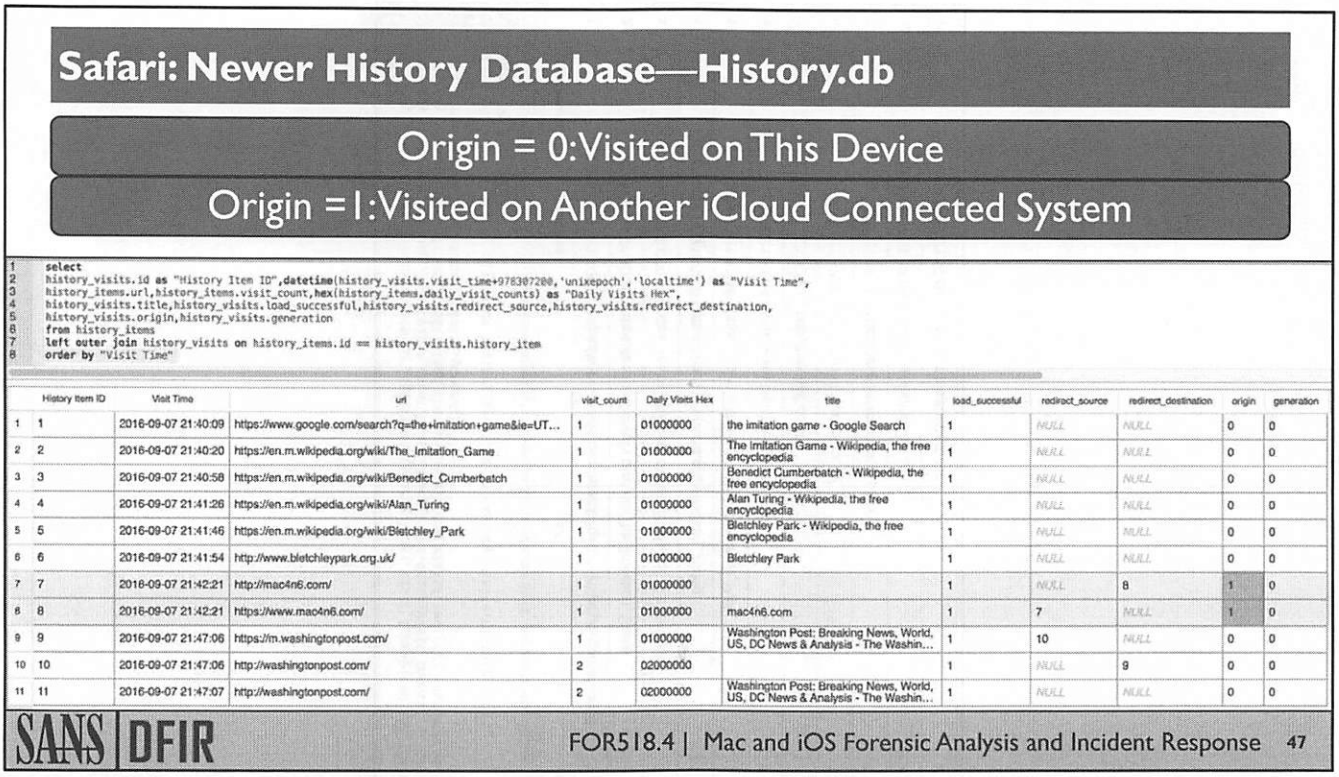
iOS Backup/FS:

- iOS 8+: `/mobile/Applications/com.apple.mobilesafari/Library/Safari/History.db`
- iOS 7: `/mobile/Applications/com.apple.mobilesafari/Library/Safari/History.plist`
- iOS 6: `/mobile/Library/Safari/History.plist`

The Safari history files changed from `History.plist` in iOS 7-/macOS 10.9- to the new SQLite database, `History.db` in iOS 8/macOS 10.10, and newer. If the user has chosen to sync their Safari history using iCloud in iOS 8+/macOS 10.10+, the history will be synced to all devices using that iCloud login. If the user clears their history on one device, it will be cleared on all devices as shown in the screenshot above.

In iOS, the history is kept for about a month, while on macOS, it is kept for potentially a year unless otherwise configured.

© 2020 Sarah Edwards



Using the SQLite query below, the following data was extracted from the Safari `History.db` file: The history ID, visit timestamp (Mac Epoch), URL, visit count, hex visit (daily), webpage title, if it loaded successfully, redirection source/destination, the origin, and generation.

Two tables in this database hold the most relevant data:

- `history_items`: Contains the URLS, domains, and visit count.
- `history_visits`: Contains the Mac Epoch timestamp of when the visits occurred, and the title of the webpage.

If iCloud Safari syncing is enabled, the "`origin`" column will display a "`1`" if that URL was viewed on another device, while a "`0`" means it was viewed from this device.

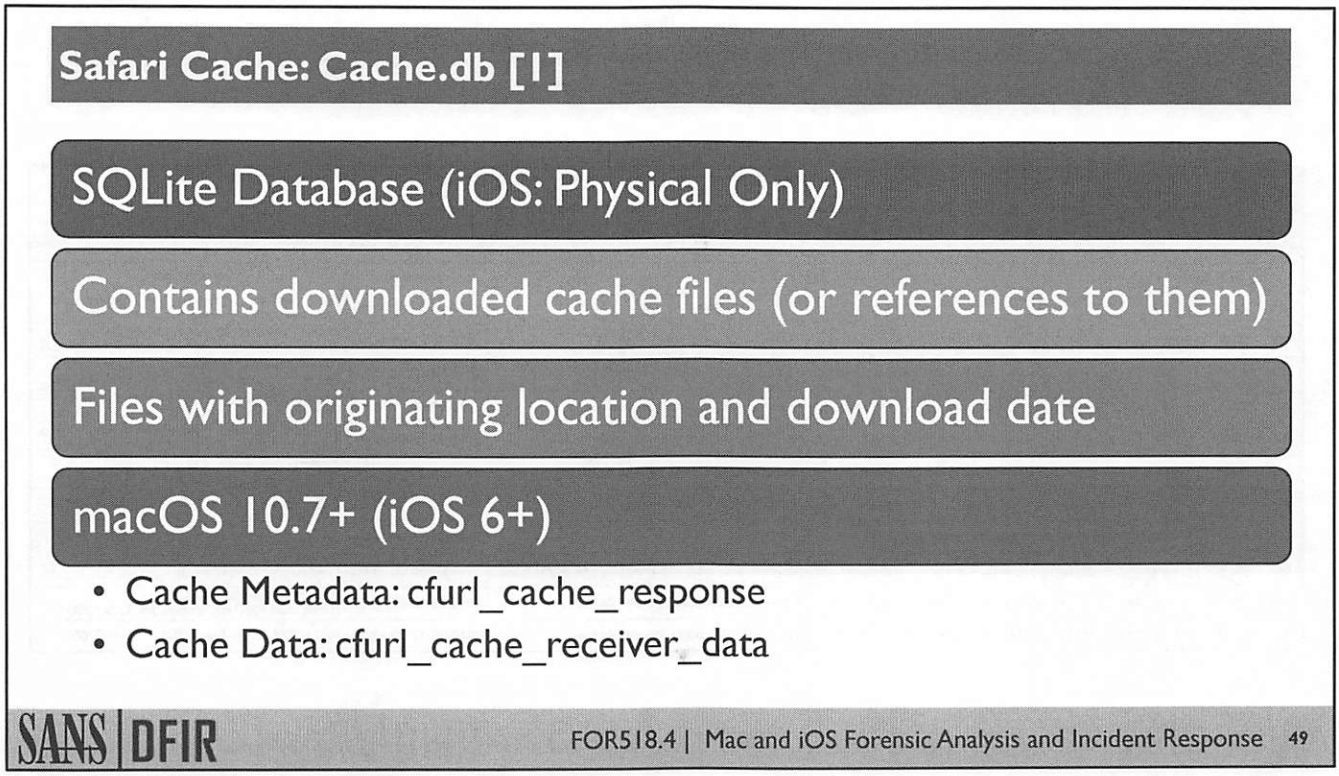
```
select
history_visits.id as "History Item ID",
datetime(history_visits.visit_time+978307200,'unixepoch','localtime') as
"Visit Time",
history_items.url,
history_items.visit_count,
history_items.daily_visit_counts,
history_items.weekly_visit_counts,
history_visits.title,
history_visits.load_successful,
history_visits.redirect_source,
history_visits.redirect_destination
from history_items
left outer join history_visits on history_items.id ==
history_visits.history_item
order by "Visit Time"
```

© 2020 Sarah Edwards

```
select
history_visits.id as "History Item ID",datetime(history_visits.visit_time+978307200,'unixepoch','localtime') as "Visit Time",
history_items.url,history_items.visit_count,hex(history_items.daily_visit_counts) as "Daily Visits Hex",
history_visits.title,history_visits.load_successful,history_visits.redirect_source,history_visits.redirect_destination,
history_visits.origin,history_visits.generation
from history_items
left outer join history_visits on history_items.id == history_visits.history_item
order by "Visit Time"
```

	History Item ID	Visit Time	url	visit_count	Daily Visits Hex	title	load_successful	redirect_source	redirect_destination	origin	generation
1	1	2016-09-07 21:40:09	https://www.google.com/search?q=the+imitation+game&ie=UT...	1	01000000	the imitation game - Google Search	1	NULL	NULL	0	0
2	2	2016-09-07 21:40:20	https://en.m.wikipedia.org/wiki/The_Imitation_Game	1	01000000	The Imitation Game - Wikipedia, the free encyclopedia	1	NULL	NULL	0	0
3	3	2016-09-07 21:40:58	https://en.m.wikipedia.org/wiki/Benedict_Cumberbatch	1	01000000	Benedict Cumberbatch - Wikipedia, the free encyclopedia	1	NULL	NULL	0	0
4	4	2016-09-07 21:41:26	https://en.m.wikipedia.org/wiki/Alan_Turing	1	01000000	Alan Turing - Wikipedia, the free encyclopedia	1	NULL	NULL	0	0
5	5	2016-09-07 21:41:46	https://en.m.wikipedia.org/wiki/Bletchley_Park	1	01000000	Bletchley Park - Wikipedia, the free encyclopedia	1	NULL	NULL	0	0
6	6	2016-09-07 21:41:54	http://www.bletchleypark.org.uk/	1	01000000	Bletchley Park	1	NULL	NULL	0	0
7	7	2016-09-07 21:42:21	http://mac4n6.com/	1	01000000		1	NULL	8	1	0
8	8	2016-09-07 21:42:21	https://www.mac4n6.com/	1	01000000	mac4n6.com	1	7	NULL	1	0
9	9	2016-09-07 21:47:06	https://m.washingtonpost.com/	1	01000000	Washington Post: Breaking News, World, US, DC News & Analysis - The Washin...	1	10	NULL	0	0
10	10	2016-09-07 21:47:06	http://washingtonpost.com/	2	02000000		1	NULL	9	0	0
11	11	2016-09-07 21:47:07	http://washingtonpost.com/	2	02000000	Washington Post: Breaking News, World, US, DC News & Analysis - The Washin...	1	NULL	NULL	0	0

© 2020 Sarah Edwards



macOS: `~/Library/Caches/com.apple.Safari/Cache.db`
iOS Physical:

- iOS 8+: `/private/var/mobile/Containers/Data/Application/<GUID>/Library/Caches/com.apple.mobilesafari/Cache.db`
- iOS 7-: `/private/var/mobile/Applications/<GUID>/Library/Caches/com.apple.mobilesafari/Cache.db`
- iOS 6-: `/private/var/mobile/Library/Caches/com.apple.mobilesafari/Cache.db`

The SQLite database `Cache.db` contains the downloaded cache files. Each file has a corresponding location and download date.

Databases for macOS 10.6 and 10.7+ systems differ slightly, but each contains a metadata table and a data table in the database.

The Safari Cache on iOS devices appears to only be available on physical acquisitions. Very often cached data is not backed up for space considerations.

© 2020 Sarah Edwards

Safari Cache: Cache.db [2]

```
1  select cfurl_cache_response.entry_ID, cfurl_cache_response.request_key, cfurl_cache_response.time_stamp,cfurl_cache_receiver_data.receiver_data
2  from cfurl_cache_response
3  left join cfurl_cache_receiver_data on cfurl_cache_response.entry_ID == cfurl_cache_receiver_data.entry_ID
```

	entry_ID	request_key	time_stamp	receiver_data
915	2046	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=mo	2016-06-02 01:11:40	["mo",["movies","money monster","movies near me","montgomery college","movie theater","m...
916	2047	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=mov	2016-06-02 01:11:43	["mov",["movies","movies near me","movie theater","moviefone","movie times","movies ou...
917	2048	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=movi	2016-06-02 01:11:43	["movi",["movies","movie times","moviefone","movietube","movie theater...
918	2049	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=movie	2016-06-02 01:11:43	["movie",["movies","movies near me","movie theater","moviefone","movie times","movies ou...
919	2050	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=movies	2016-06-02 01:11:43	["movies",["movies","movies near me","movies in theaters","movies 2016","movies 2015","movie...
920	2051	https://dalk4zrp4jp3q.cloudfront.net/sprites/6ee1bbb017b910abbc9edb065806bed0_iphone_2x.json	2016-06-16 00:13:58	{"iphone_AppleHelp_iOS_2x":[1240,1138,120,120],"iphone_MLB_2x":[920,6...
921	2052	https://dalk4zrp4jp3q.cloudfront.net/sprites/6ee1bbb017b910abbc9edb065806bed0_iphone_2x.png	2016-06-16 00:13:58	6A38E58D-8D16-49BD-B996-8D259D3ED7CA
922	2053	https://github.com/apple-touch-icon-180x180.png	2016-06-16 00:13:58	EA26805A-DD2B-4608-9BD7-D0F6013888CE
923	2054	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=JetBlue+Airways+1055	2016-06-16 00:14:02	["JetBlue Airways 1055",[]]
924	2055	https://dalk4zrp4jp3q.cloudfront.net/sprites/767a2e84bda337de2c375557a28e6c5b_iphone_2x.json	2016-06-17 22:42:36	{"iphone_AppleHelp_iOS_2x":[1240,1138,120,120],"iphone_MLB_2x":[920,6...
925	2056	https://dalk4zrp4jp3q.cloudfront.net/sprites/767a2e84bda337de2c375557a28e6c5b_iphone_2x.png	2016-06-17 22:42:36	F2D17B7A-B05B-4924-82CE-7E0484C76B98

The SQLite query below extracts and correlates the cache from two tables: One metadata, the other with data. The `cfurl_cache_response` table contains cache file metadata, including the file cached and its corresponding timestamp. The `cfurl_cache_receiver_data` table contains the cached file. The cached file can be matched up with its metadata by using the `entry_ID` number.

Each cached entry has an associated entry_ID, URL (`request_key`), timestamp, and cached data (`receiver_data`). The `receiver_data` column may contain a GUID that can be associated with a file in the `/Caches` and `/fsCachedData` directory, shown next.

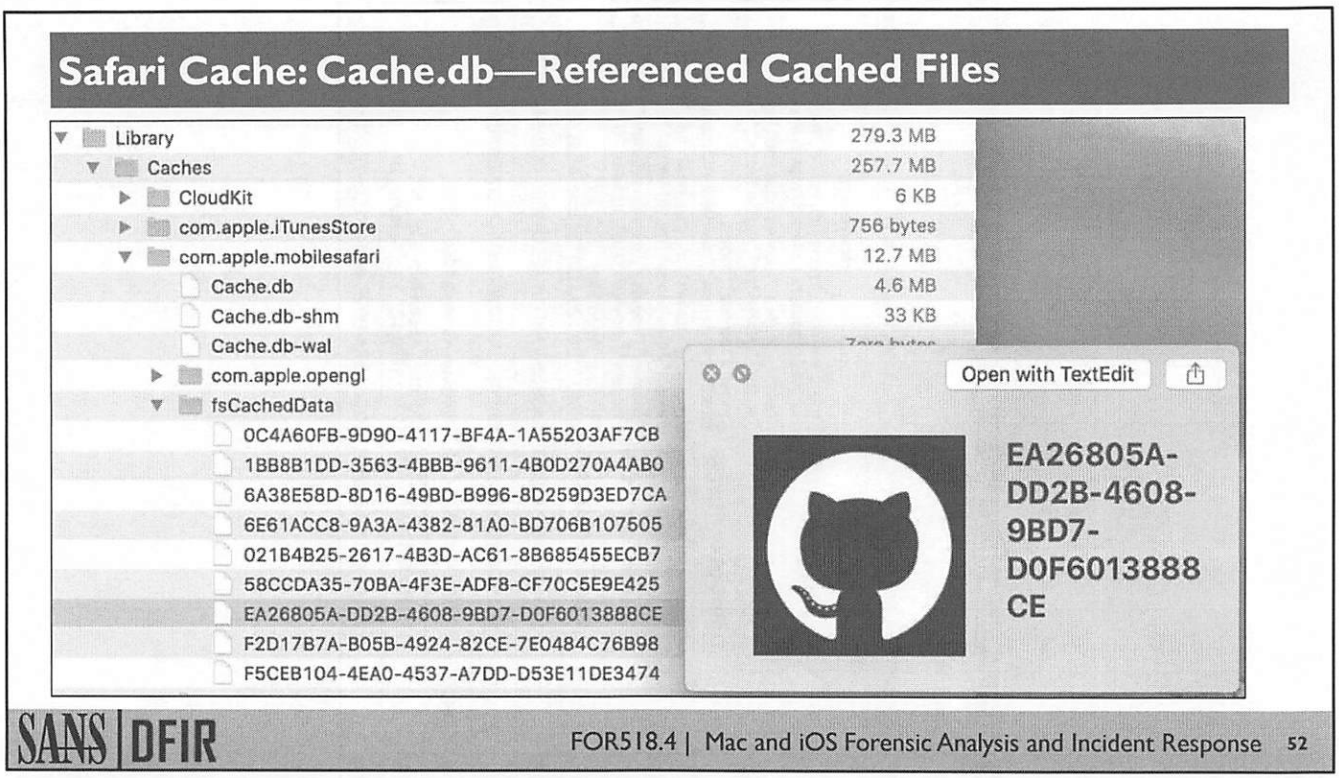
```
select
cfurl_cache_response.entry_ID,
cfurl_cache_response.request_key,
cfurl_cache_response.time_stamp,
cfurl_cache_receiver_data.receiver_data
from cfurl_cache_response
left join cfurl_cache_receiver_data on cfurl_cache_response.entry_ID ==
cfurl_cache_receiver_data.entry_ID
```


© 2020 Sarah Edwards

```
1  select cfurl_cache_response.entry_ID, cfurl_cache_response.request_key, cfurl_cache_response.time_stamp,cfurl_cache_receiver_data.receiver_data
2  from cfurl_cache_response
3  left join cfurl_cache_receiver_data on cfurl_cache_response.entry_ID == cfurl_cache_receiver_data.entry_ID
```

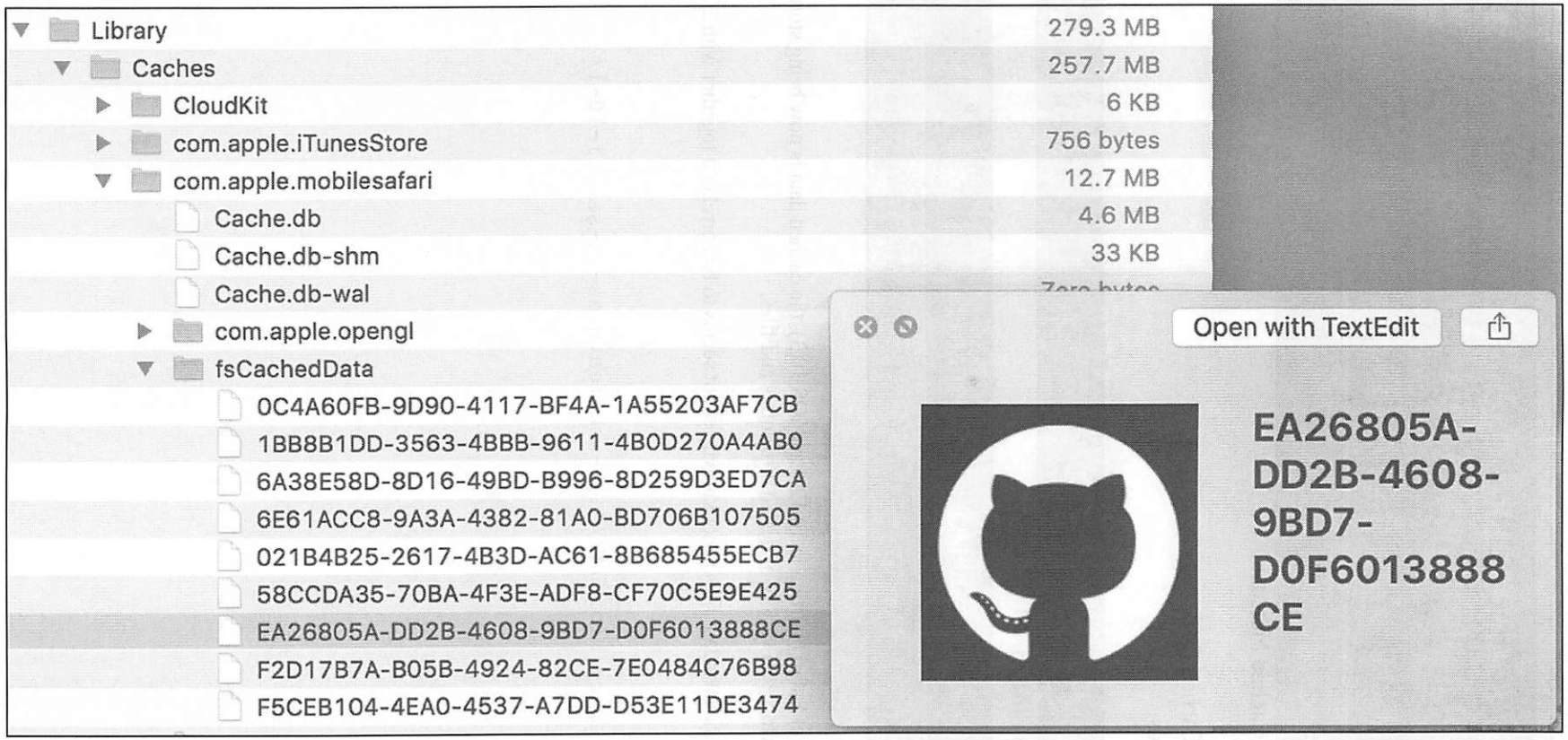
	entry_ID	request_key	time_stamp	receiver_data
915	2046	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=mo	2016-06-02 01:11:40	["mo",["movies","money monster","movies near me","montgomery college","movie theater","m…
916	2047	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=mov	2016-06-02 01:11:43	["mov",["movies","movies near me","movie theater","moviefone","movie times","movies ou…
917	2048	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=movi	2016-06-02 01:11:43	["movi",["movies","movie times","moviefone","movietube","movie theater…
918	2049	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=movie	2016-06-02 01:11:43	["movie",["movies","movies near me","movie theater","moviefone","movie times","movies ou…
919	2050	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=movies	2016-06-02 01:11:43	["movies",["movies","movies near me","movies in theaters","movies 2016","movies 2015","movie…
920	2051	https://dalk4zrp4jp3q.cloudfront.net/sprites/6ee1bbb017b910abbc9edb065806bed0_iphone_2x.json	2016-06-16 00:13:58	{"iphone_AppleHelp_iOS_2x":[1240,1138,120,120],"iphone_MLB_2x":[920,6…
921	2052	https://dalk4zrp4jp3q.cloudfront.net/sprites/6ee1bbb017b910abbc9edb065806bed0_iphone_2x.png	2016-06-16 00:13:58	6A38E58D-8D16-49BD-B996-8D259D3ED7CA
922	2053	https://github.com/apple-touch-icon-180x180.png	2016-06-16 00:13:58	EA26805A-DD2B-4608-9BD7-D0F6013888CE
923	2054	https://clients1.google.com/complete/search?json=t&nolabels=t&client=iphonesafari&q=JetBlue+Airways+1055	2016-06-16 00:14:02	["JetBlue Airways 1055",[]]
924	2055	https://dalk4zrp4jp3q.cloudfront.net/sprites/767a2e84bda337de2c375557a28e6c5b_iphone_2x.json	2016-06-17 22:42:36	{"iphone_AppleHelp_iOS_2x":[1240,1138,120,120],"iphone_MLB_2x":[920,6…
925	2056	https://dalk4zrp4jp3q.cloudfront.net/sprites/767a2e84bda337de2c375557a28e6c5b_iphone_2x.png	2016-06-17 22:42:36	F2D17B7A-B05B-4924-82CE-7E0484C76B98

© 2020 Sarah Edwards

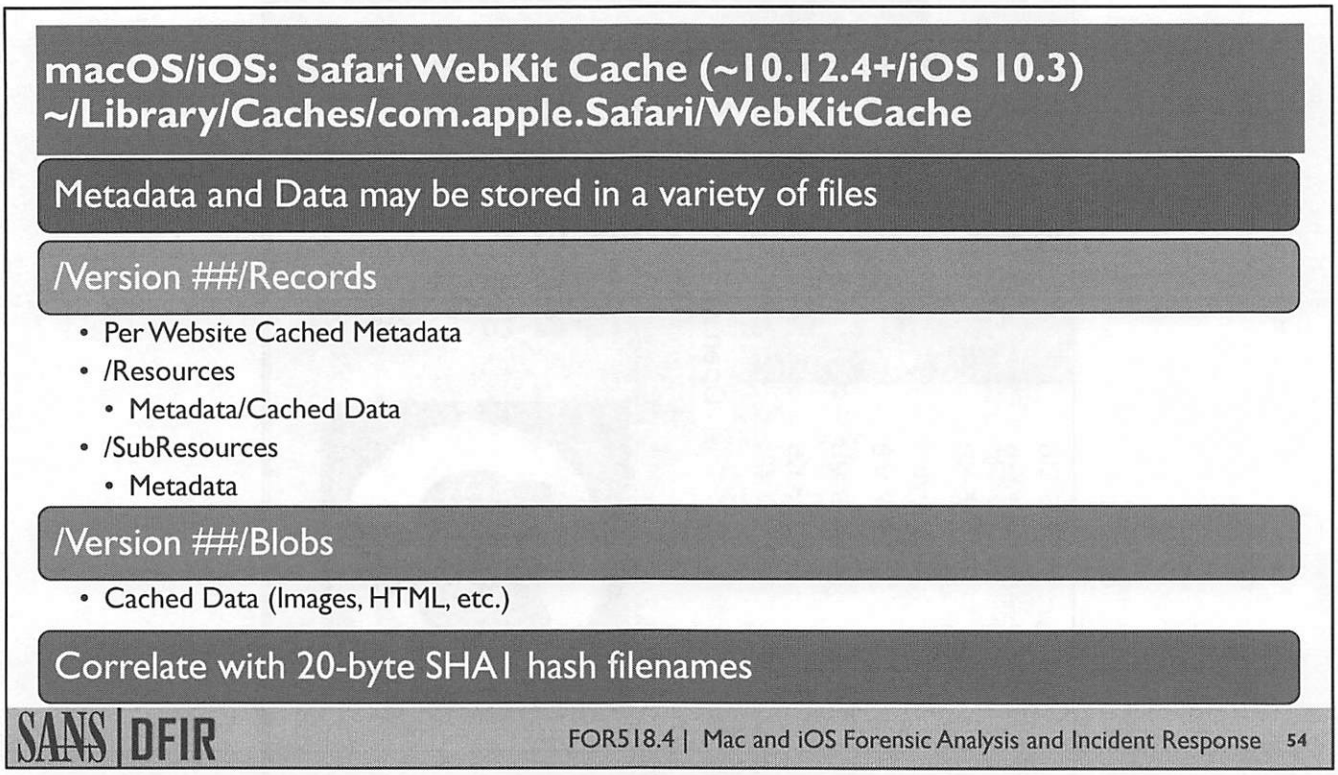


The externally referenced cached files will be located in the `fsCachedData` directory, as shown above. The filenames are the same as the GUID in the Safari `Cache.db` database file.

© 2020 Sarah Edwards



© 2020 Sarah Edwards



On newer versions of Safari (starting in 10.12.4 and iOS 10.3), the cached data is now being stored in the WebKitCache directory in a variety of nested directories and files.

The true format of the metadata files is not known yet but seems to be correlated together with 20-byte filename hashes.

Reference:
`https://webkit.org/blog/7477/new-web-features-in-safari-10-1/`

© 2020 Sarah Edwards

Safari: Saved Session Data—macOS LastSession.plist

▼ SessionWindows	Array	(1 item)
▼ Item 0	Dictionary	(14 items)
SelectedTabIndex	Number	0
TabBarHidden	Boolean	NO
DateClosed	Date	Feb 8, 2017, 10:23:40 PM
FavoritesBarHidden	Boolean	YES
IsPopupWindow	Boolean	NO
PrefersReadingListSidebarVisible	Boolean	NO
FullScreenFavoritesBarHidden	Boolean	NO
WindowStateVersion	String	2.0
WindowUUID	String	3638500E-48C8-4F21-9B17-7C355FE278AB
Miniaturized	Boolean	NO
▼ TabStates	Array	(4 items)
▼ Item 0	Dictionary	(14 items)
IsDisposable	Boolean	NO
SessionState	Data	<9a6f7a80 24aad462 4ac9ff4a 2ed6d01d 15fe8
▼ AncestorTabIdentifiers	Array	(0 items)
DateClosed	Date	Feb 8, 2017, 10:23:40 PM
SessionStateIsEncrypted	Boolean	YES
TabIndex	Number	0
WindowUUID	String	3638500E-48C8-4F21-9B17-7C355FE278AB
LastVisitTime	Number	508,301,378.142209
TabUUID	String	20DF813F-6392-4B8B-80F3-D5CAD8AA51DF
TabURL	String	https://en.wikipedia.org/wiki/Bletchley_Park
TabIdentifier	Number	9
TabTitle	String	Bletchley Park - Wikipedia
ProcessIdentifier	Number	27,904
IsMuted	Boolean	NO

macOS: `~/Library/Safari/LastSession.plist`

The `LastSession.plist` contains items from the last browsing session. If multiple browsing tabs were open, there will be multiple items showing the visited URLs.

Each tab will have a tab identifier, which will not necessarily be in numeric order if tabs were removed. The `TabTitle` and `TabURL` hold the webpage title and URL, respectively. The `SessionState` key may hold more information in older versions, shown on the next slide. However, since 10.10, this data is now encrypted.

On iOS, the `SuspendState.plist` file is similar to the `LastSession.plist` file on macOS: It keeps track of the opened tabs. The `SafariStateDocuments` key contains a separate item for each tab the user has open. The `SafariStateDocumentSessionState` key contains the tab history in an embedded plist file. The `SafariStatePrivateDocuments` key contains the tab information for tabs in "Private Mode".

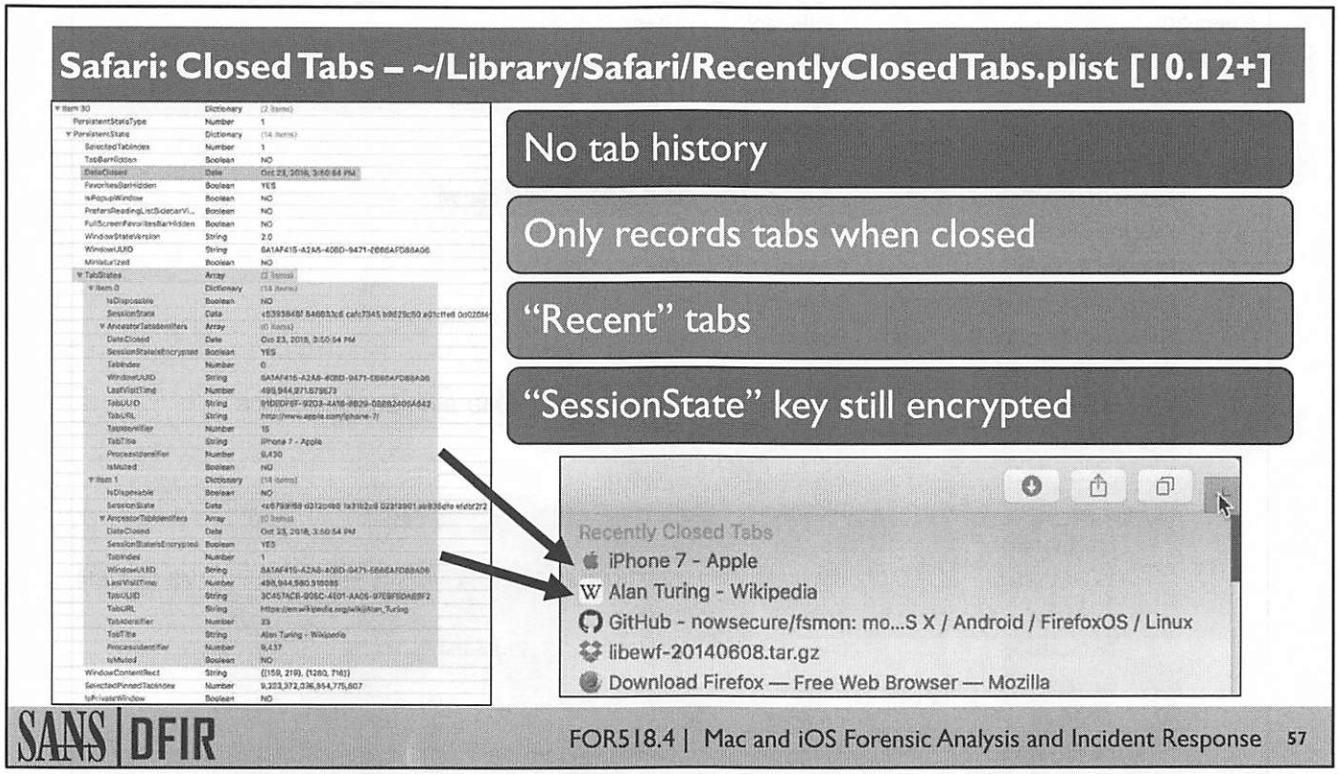
Physical: iOS 8+:
`/private/var/mobile/Containers/Data/Application/<GUID>/Library/Safari/SuspendState.plist`
Backup/FS: iOS 7+:
`/mobile/Applications/com.apple.mobilesafari/Library/Safari/SuspendState.plist`

▼ Root	Dictionary	(5 items)
SafariStateActiveDocumentIndex	Number	0
▼ SafariStateDocuments	Array	(1 item)
▼ Item 0	Dictionary	(13 items)
SafariStateDocumentUsesPrivateBrowsingStyle	Boolean	NO
SafariStateDocumentLastRenderTreeSize	Number	0
SafariStateDocumentShouldRestoreReader	Boolean	NO
SafariStateDocumentLastViewedTime	Number	494,992,859.421361
SafariStateDocumentBookmark	Number	0
SafariStateDocumentWasOpenedFromLink	Boolean	NO
SafariStateDocumentURL	String	https://m.washingtonpost.com/
SafariStateDocumentUserVisibleURL	String	https://m.washingtonpost.com/
SafariStateDocumentUUID	String	A81ED39F-6DD4-4B88-8CC2-71B89AD2FE80
SafariStateDocumentTitle	String	Washington Post: Breaking News, World, US, DC News & Analysis - The Washington Post
SafariStateDocumentReaderViewTopScrollOffset	Number	0
SafariStateDocumentSessionState	Data	<00000002 62706c69 73743030 d2010203 045e5265 6e646572 54726565 53697a65
SafariStateDocumentDisplayingStandaloneImage	Boolean	NO
▶ SafariStatePrivateDocuments	Array	(2 items)
SafariStatePrivateActiveDocumentIndex	Number	1
SafariStateFileVersion	Number	1

© 2020 Sarah Edwards

▽ Root	Dictionary	(2 items)
SessionVersion	String	1.0
▽ SessionWindows	Array	(1 item)
▽ Item 0	Dictionary	(10 items)
WindowStateVersion	String	2.0
LocationBarHidden	Boolean	NO
▽ TabStates	Array	(2 items)
▽ Item 0	Dictionary	(5 items)
▽ AncestorTabIdentifers	Array	(0 items)
TabIdentifier	Number	1
TabTitle	String	Apple - Start
TabURL	String	http://www.apple.com/startpage/
SessionState	Data	<00000002 62706c69 73743030 d101025e 53657373 696f6e48 6973746f 7279d203 04(
▽ Item 1	Dictionary	(5 items)
▽ AncestorTabIdentifers	Array	(0 items)
TabIdentifier	Number	8
TabTitle	String	elephants - Google Search
TabURL	String	https://www.google.com/search?client=safari&rls=en&q=elephants&ie=UTF-8&oe=UTF-8
SessionState	Data	<00000002 62706c69 73743030 d101025e 53657373 696f6e48 6973746f 7279d203 04(
Miniaturized	Boolean	NO
SelectedTabIndex	Number	1
TabBarHidden	Boolean	NO
PrefersReadingListSidebarVisible	Boolean	NO
WindowContentRect	String	{{0, 4}, {1366, 691}}
FavoritesBarHidden	Boolean	NO
StatusBarHidden	Boolean	YES

© 2020 Sarah Edwards



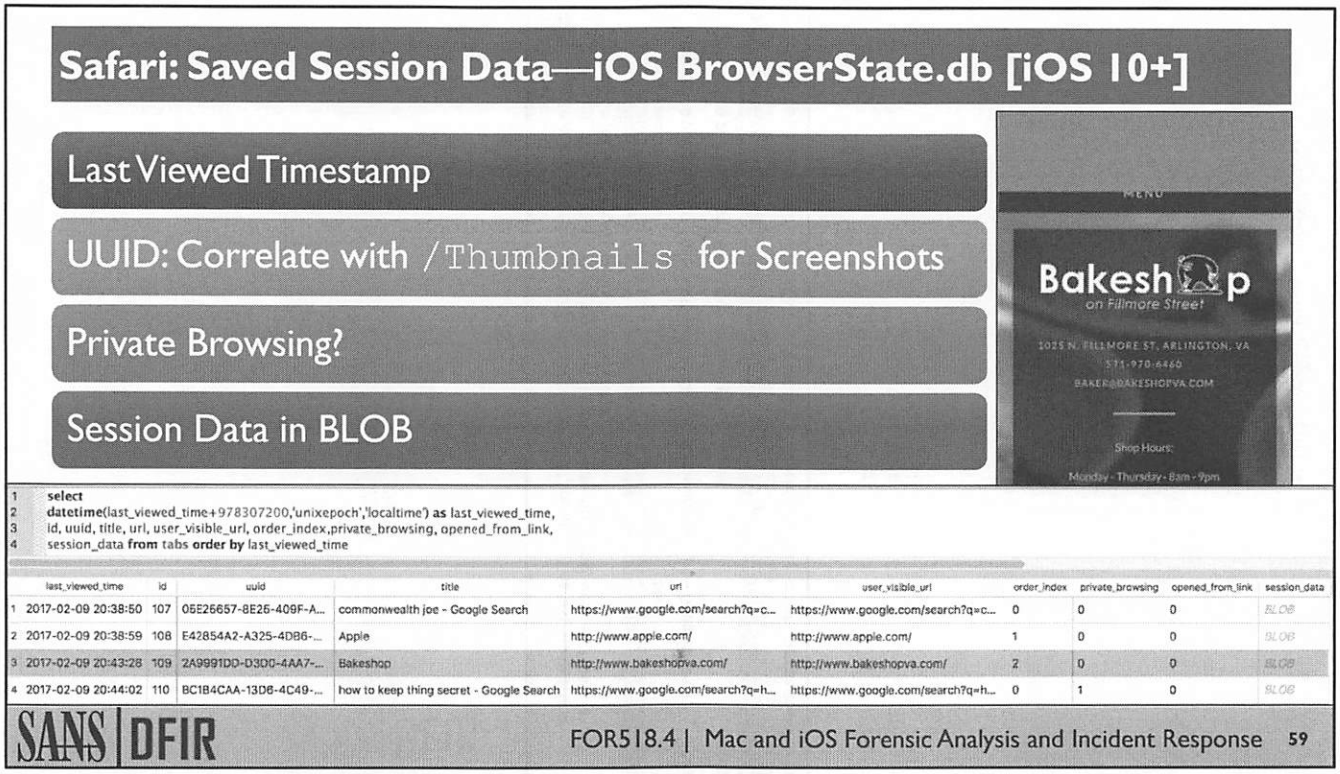
A new plist with 10.12 is `RecentlyClosedTabs.plist`, which appears very similar to `LastSession.plist`; however, it keeps a "recent" history.

Along with newer versions of `LastSession.plist`, the `SessionState` key is still encrypted. This plist only keeps track of the current visible tabs that the user had opened, and not the previous history within the tab itself.

© 2020 Sarah Edwards

▼ Item 30	Dictionary	(2 items)
PersistentStateType	Number	1
▼ PersistentState	Dictionary	(14 items)
SelectedTabIndex	Number	1
TabBarHidden	Boolean	NO
DateClosed	Date	Oct 23, 2016, 3:50:54 PM
FavoritesBarHidden	Boolean	YES
IsPopupWindow	Boolean	NO
PrefersReadingListSidebarVi...	Boolean	NO
FullScreenFavoritesBarHidden	Boolean	NO
WindowStateVersion	String	2.0
WindowUUID	String	8A1AF415-A2A8-40BD-9471-E666AFD88A06
Miniaturized	Boolean	NO
▼ TabStates	Array	(2 items)
▼ Item 0	Dictionary	(14 items)
IsDisposable	Boolean	NO
SessionState	Data	<5393848f 846633c6 cafc7345 b9825c50 a01cffe8 0d026f4
▼ AncestorTabIdentifers	Array	(0 items)
DateClosed	Date	Oct 23, 2016, 3:50:54 PM
SessionStateIsEncrypted	Boolean	YES
TabIndex	Number	0
WindowUUID	String	8A1AF415-A2A8-40BD-9471-E666AFD88A06
LastVisitTime	Number	498,944,971.675673
TabUUID	String	91DEDF6F-92D3-4A16-8B29-0BBB2405A642
TabURL	String	http://www.apple.com/iphone-7/
TabIdentifier	Number	15
TabTitle	String	iPhone 7 - Apple
ProcessIdentifier	Number	9,430
IsMuted	Boolean	NO
▼ Item 1	Dictionary	(14 items)
IsDisposable	Boolean	NO
SessionState	Data	<c6799f68 d312c4b5 1a31b2c8 023f3901 ab835dfe efdbf2f2
▼ AncestorTabIdentifers	Array	(0 items)
DateClosed	Date	Oct 23, 2016, 3:50:54 PM
SessionStateIsEncrypted	Boolean	YES
TabIndex	Number	1
WindowUUID	String	8A1AF415-A2A8-40BD-9471-E666AFD88A06
LastVisitTime	Number	498,944,980.518085
TabUUID	String	3C457AC6-905C-4E01-AA05-97E9F5DAB9F2
TabURL	String	https://en.wikipedia.org/wiki/Alan_Turing
TabIdentifier	Number	23
TabTitle	String	Alan Turing - Wikipedia
ProcessIdentifier	Number	9,437
IsMuted	Boolean	NO
WindowContentRect	String	{{159, 219}, {1280, 716}}
SelectedPinnedTabIndex	Number	9,223,372,036,854,775,807
IsPrivateWindow	Boolean	NO

© 2020 Sarah Edwards



iOS 10 Physical:
`/private/var/mobile/Containers/Data/Application/<GUID>/Library/Safari/BrowserState.db`
iOS 10 Backup:
`/mobile/Applications/com.apple.mobilesafari/Library/Safari/BrowserState.db`

Like other files that have changed from plists to databases, there is a new format for iOS 10. The `BrowserState.db` file is a SQLite database that contains relatively the same information as the previous plist files for the browser session data.

The screenshot above shows the last viewed time for the tab contents (in Mac Epoch), tab identifiers, tab metadata, and browser visit information. There are a few more metadata items that are not shown that may be of use to the analyst.

```
select
datetime(last_viewed_time+978307200,'unixepoch','localtime') as
last_viewed_time,
id,
uuid,
title,
url,
user_visible_url,
order_index,
private_browsing,
opened_from_link,
session_data from tabs order by last_viewed_time
```

© 2020 Sarah Edwards

```
1  select
2  datetime(last_viewed_time+978307200,'unixepoch','localtime') as last_viewed_time,
3  id, uuid, title, url, user_visible_url, order_index,private_browsing, opened_from_link,
4  session_data from tabs order by last_viewed_time
```

	last_viewed_time	id	uuid	title	url	user_visible_url	order_index	private_browsing	opened_from_link	session_data
1	2017-02-09 20:38:50	107	05E25657-8E25-409F-A...	commonwealth joe - Google Search	https://www.google.com/search?q=c...	https://www.google.com/search?q=c...	0	0	0	BLOB
2	2017-02-09 20:38:59	108	E42B54A2-A325-4DB6-...	Apple	http://www.apple.com/	http://www.apple.com/	1	0	0	BLOB
3	2017-02-09 20:43:28	109	2A9991DD-D3D0-4AA7-...	Bakeshop	http://www.bakeshopva.com/	http://www.bakeshopva.com/	2	0	0	BLOB
4	2017-02-09 20:44:02	110	BC1B4CAA-13D6-4C49-...	how to keep thing secret - Google Search	https://www.google.com/search?q=h...	https://www.google.com/search?q=h...	0	1	0	BLOB

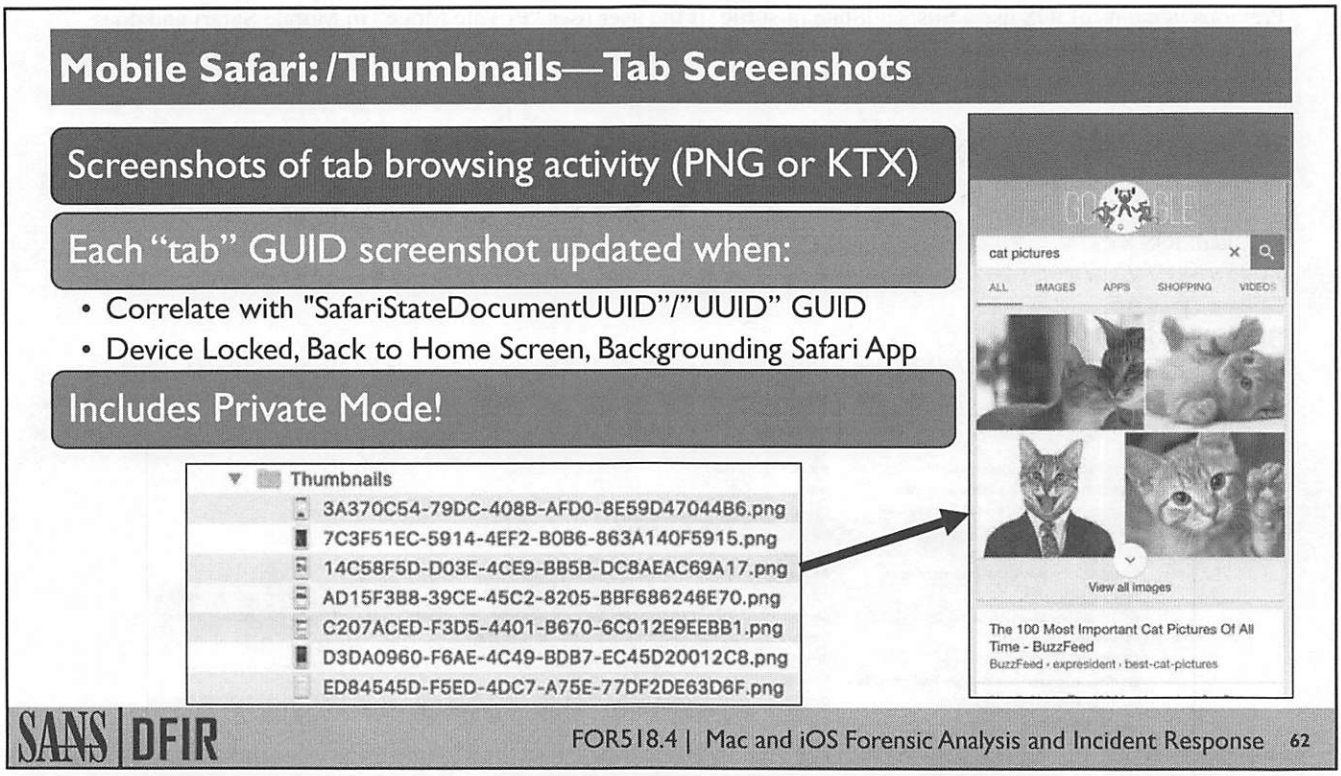
© 2020 Sarah Edwards

Previous versions of iOS use a SuspendState.plist file. If the user uses "Private Mode" in Mobile Safari and does not close the tab after browsing, the data shown is still available in the `SuspendState.plist` file. The data is not cached in the Safari Cache or in the `History.db` files, however.

The last viewed time, URL, and webpage title are recorded, as well as possible `SessionState` tab histories.

Physical: iOS 8+:
`/private/var/mobile/Containers/Data/Application/<GUID>/Library/Safari/SuspendState.plist`
Backup/FS: iOS 7+:
`/mobile/Applications/com.apple.mobilesafari/Library/Safari/SuspendState.plist`

▼ SafariStatePrivateDocuments	Array	(2 items)
▼ Item 0	Dictionary	(13 items)
SafariStateDocumentUsesPrivateBrowsingStyle	Boolean	YES
SafariStateDocumentLastRenderTreeSize	Number	0
SafariStateDocumentShouldRestoreReader	Boolean	NO
SafariStateDocumentLastViewedTime	Number	494,992,920.13129
SafariStateDocumentBookmark	Number	0
SafariStateDocumentWasOpenedFromLink	Boolean	NO
SafariStateDocumentURL	String	https://www.google.com/search?q=secret+stuff&ie=UTF-8&oe=UTF-8&hl=en-us&client=safari
SafariStateDocumentUserVisibleURL	String	https://www.google.com/search?q=secret+stuff&ie=UTF-8&oe=UTF-8&hl=en-us&client=safari
SafariStateDocumentUUID	String	AD15F3B8-39CE-45C2-8205-BBF686246E70
SafariStateDocumentTitle	String	secret stuff - Google Search
SafariStateDocumentReaderViewTopScrollOffset	Number	0
SafariStateDocumentSessionState	Data	<00000002 62706c69 73743030 d2010203 045e5265 6e646572 54726565 53697a65 5e5365
SafariStateDocumentDisplayingStandaloneImage	Boolean	NO
▼ Item 1	Dictionary	(13 items)
SafariStateDocumentUsesPrivateBrowsingStyle	Boolean	YES
SafariStateDocumentLastRenderTreeSize	Number	0
SafariStateDocumentShouldRestoreReader	Boolean	NO
SafariStateDocumentLastViewedTime	Number	494,992,920.131291
SafariStateDocumentBookmark	Number	0
SafariStateDocumentWasOpenedFromLink	Boolean	NO
SafariStateDocumentURL	String	https://www.google.com/search?q=cat+pictures&ie=UTF-8&oe=UTF-8&hl=en-us&client=safari
SafariStateDocumentUserVisibleURL	String	https://www.google.com/search?q=cat+pictures&ie=UTF-8&oe=UTF-8&hl=en-us&client=safari
SafariStateDocumentUUID	String	14C58F5D-D03E-4CE9-BB5B-DC8AEAC69A17
SafariStateDocumentTitle	String	cat pictures - Google Search
SafariStateDocumentReaderViewTopScrollOffset	Number	0
SafariStateDocumentSessionState	Data	<00000002 62706c69 73743030 d2010203 045e5265 6e646572 54726565 53697a65 5e5365
SafariStateDocumentDisplayingStandaloneImage	Boolean	NO



Physical:

- iOS 8+: `/private/var/mobile/Containers/Data/Application/<GUID>/Library/Safari/Thumbnails/`
- iOS 7-: `/private/var/mobile/Applications/<GUID>/Library/Safari/Thumbnails/`
- iOS 6-: `/mobile/Library/Caches/com.apple.mobilesafari/Thumbnails/`

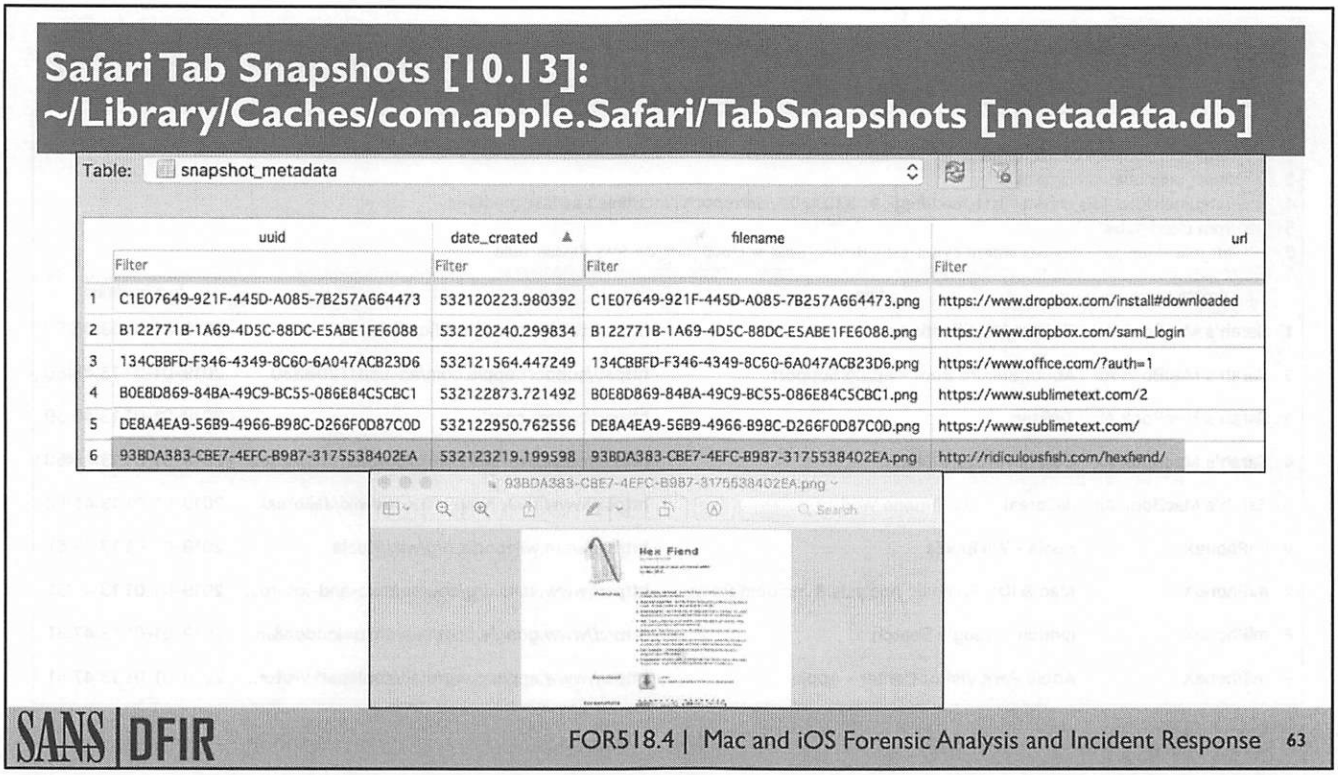
Backup/FS:

- iOS 7+: `/mobile/Applications/com.apple.mobilesafari/Library/Safari/Thumbnails/`
- iOS 6-: `/mobile/Library/Caches/com.apple.mobilesafari/Thumbnails/`

Each GUID in the Thumbnails directory contains a thumbnail of the tab "screenshot", including those in Private Mode. If the user visits another website, the tab "screenshot" will get updated with a new "screenshot". This happens when the Safari application is backgrounded, the device is locked, or the user selects a different tab.

In the example above, this device has seven tabs open, some of which are in "Private Mode".

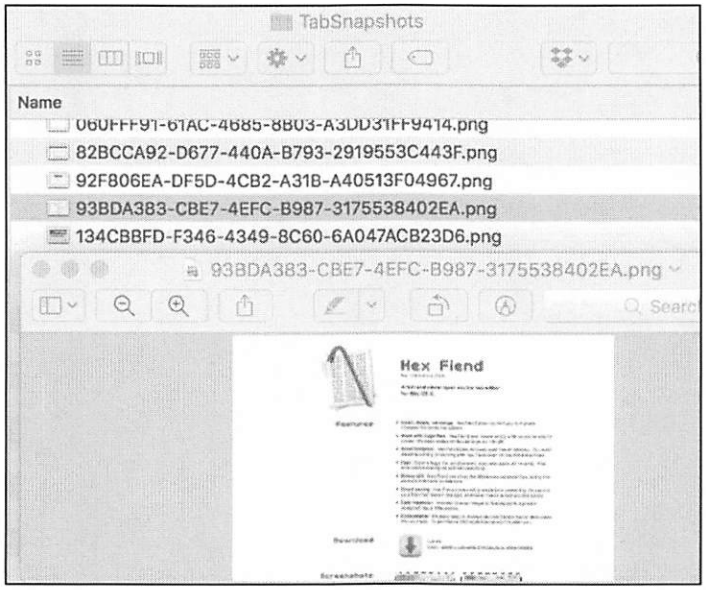
© 2020 Sarah Edwards



Previous tab screenshots may be found in the 10.13 Safari cache directory. These screenshots are not exactly high resolution; however, we can determine what website they are a screenshot of by looking in the related `metadata.db` SQLite database file.

In the example above, each tab screenshot is assigned a UUID, which is also used in the saved filename. Going to the `TabSnapshots` directory, we can open and view these screenshots.

Older versions of macOS may have the directory, ~/Library/Caches/com.apple.Safari/Webpage Previews/. These snapshots only appear to be created if the user selected the "Top Sites" button.



macOS/iOS – Cloud Tabs – CloudTabs.db

```
1  select
2  cloud_tab_devices.device_name,
3  cloud_tabs.title, cloud_tabs.url,
4  datetime(cloud_tab_devices.last_modified+978307200,'unixepoch','localtime') as last_modified
5  from cloud_tabs
6  left join cloud_tab_devices where cloud_tabs.device_uuid = cloud_tab_devices.device_uuid
```

	device_name	title	url	last_modified
1	Sarah's MacBook Air	Twitter / Notifications	https://twitter.com/i/notifications	2019-01-01 13:45:50
2	Sarah's MacBook Air	About Secure Boot - Apple Support	https://support.apple.com/en-us/HT208330	2019-01-01 13:45:50
3	Sarah's MacBook Air	Twitter	https://twitter.com/	2019-01-01 13:45:50
4	Sarah's MacBook Air	Objective-See's Blog	https://objective-see.com/blog/blog_0x3C.html	2019-01-01 13:45:50
5	Sarah's MacBook Air	Jailbreak - The iPhone Wiki	https://www.theiphonewiki.com/wiki/Jailbreak	2019-01-01 13:45:50
6	miPhoneX	Koala - Wikipedia	https://en.m.wikipedia.org/wiki/Koala	2019-01-01 13:47:51
7	miPhoneX	Mac & IOS Forensic Analysis & Incident Resp...	https://www.sans.org/course/mac-and-ios-fo...	2019-01-01 13:47:51
8	miPhoneX	london - Google Search	https://www.google.com/search?q=london&ie...	2019-01-01 13:47:51
9	miPhoneX	Apple Park Visitor Center - Apple	https://www.apple.com/retail/appleparkvisitor...	2019-01-01 13:47:51

SANS | DFIR FOR518.4 | Mac and iOS Forensic Analysis and Incident Response 64

macOS: ~/Library/Safari/CloudTabs.db
iOS: [/private/var/]mobile/Library/Safari/CloudTabs.db

Devices that are syncing a user's Safari web history can be found in the CloudTabs.db database.

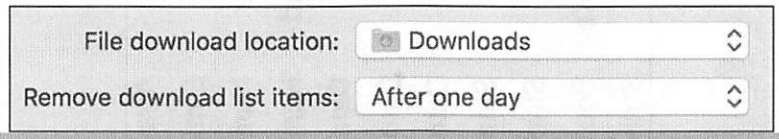
The example above shows two devices by hostname: Sarah's MacBook Air and miPhoneX. Each device has a few (non-private) tabs open in their respective Safari browsers. The last modification timestamp is a good place to see how stale these tabs are.

© 2020 Sarah Edwards

Safari: macOS Downloads – ~/Library/Safari/Downloads.plist

▼ DownloadHistory	Array	(1 item)
▼ Item 0	Dictionary	(9 items)
DownloadEntryProgressBytesSoFar	Number	66,812,133
DownloadEntryProgressTotalToLoad	Number	66,812,133
DownloadEntryBookmarkBlob	Data	<626f6f6b 70030000 00000410 30000000 00000000 00000000 000000
DownloadEntryDateAddedKey	Date	Dec 14, 2014, 4:51:26 PM
DownloadEntryDateFinishedKey	Date	Dec 14, 2014, 4:59:07 PM
DownloadEntryIdentifier	String	01856613-E26D-4488-B382-98A2F561782F
DownloadEntryURL	String	https://dl.google.com/chrome/mac/stable/GGRM/googlechrome.dmg
DownloadEntryRemoveWhenDoneKey	Boolean	NO
DownloadEntryPath	String	~/Downloads/googlechrome.dmg

- 10.12+: Default—Remove Downloads after 1 Day
  - Removed from `Downloads.plist`



The Safari directory contains the `Downloads.plist` file. This file contains many items that may interest a forensic analyst about the Safari download history:

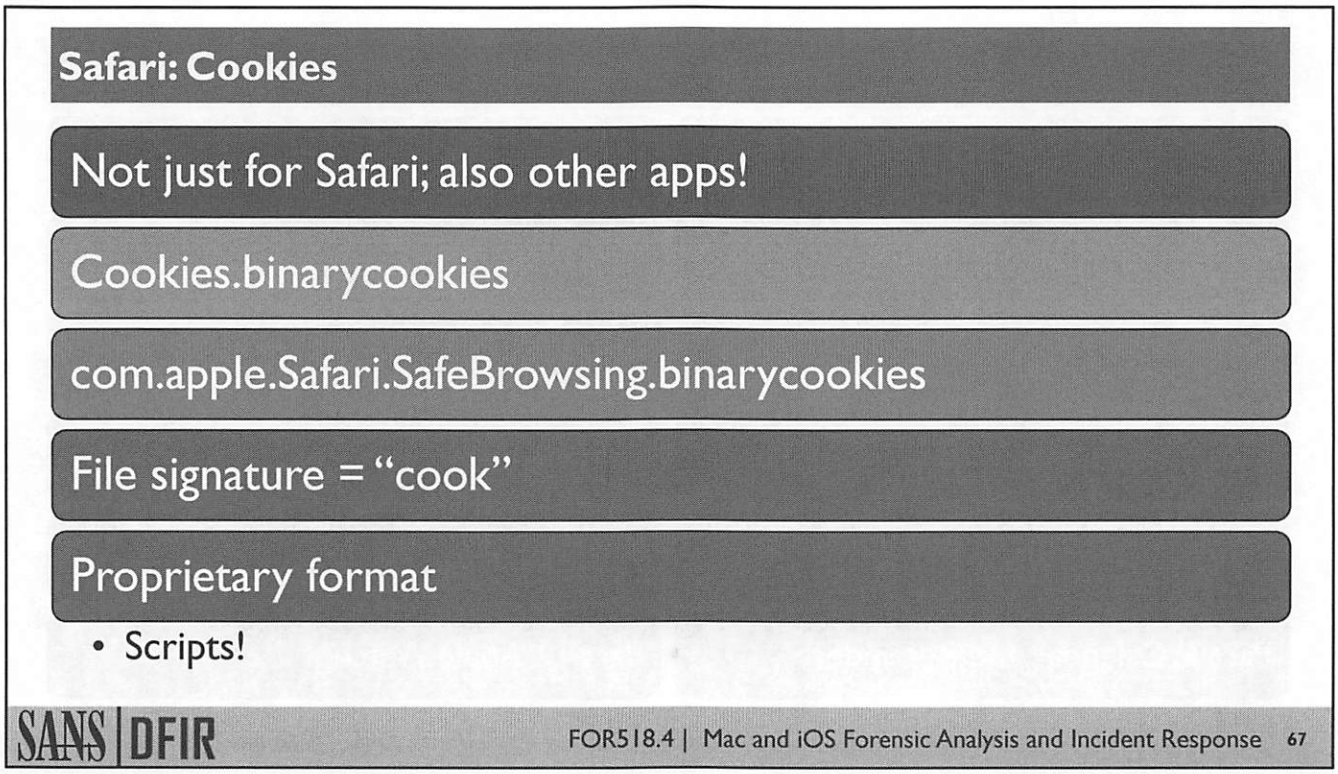
- **DownloadEntryIdentifier:** Each download has a unique GUID
- **DownloadEntryURL:** A URL is saved showing where the download originated
- **DownloadEntryProgressTotalToLoad, ProgressBytesSoFar**: These keys store the total bytes and downloaded bytes for the download. The downloads file may not always have the same quantity of bytes if the download was canceled or otherwise stopped.
- **DownloadEntryPath:** Where the item was downloaded to; very likely will be in the default Downloads directory (`~/Downloads`)
- **DownloadEntryBookmark/Alias Blob:** Bookmark or Alias data (Aliases are used in 10.6 and 10.7)
- **DownloadEntryDateAddedKey, DownloadEntryDateFinishedKey:** The timestamps of when the download started and finished (10.10+)

10.12 introduced a new default; instead of having the user manually remove items from the Safari Download list, they have instituted a "`Remove download list items`" configuration item that has "`After one day`" selected by default. Other options include: "`When Safari Quits`", "`Upon Successful Download`", or "`Manually`".

© 2020 Sarah Edwards

▼ DownloadHistory	Array	(1 item)
▼ Item 0	Dictionary	(9 items)
DownloadEntryProgressBytesSoFar	Number	66,812,133
DownloadEntryProgressTotalToLoad	Number	66,812,133
DownloadEntryBookmarkBlob	Data	<626f6f6b 70030000 00000410 30000000 00000000 00000000 000000
DownloadEntryDateAddedKey	Date	Dec 14, 2014, 4:51:26 PM
DownloadEntryDateFinishedKey	Date	Dec 14, 2014, 4:59:07 PM
DownloadEntryIdentifier	String	01856613-E26D-4488-B382-98A2F561782F
DownloadEntryURL	String	https://dl.google.com/chrome/mac/stable/GGRM/googlechrome.dmg
DownloadEntryRemoveWhenDoneKey	Boolean	NO
DownloadEntryPath	String	~/Downloads/googlechrome.dmg

© 2020 Sarah Edwards

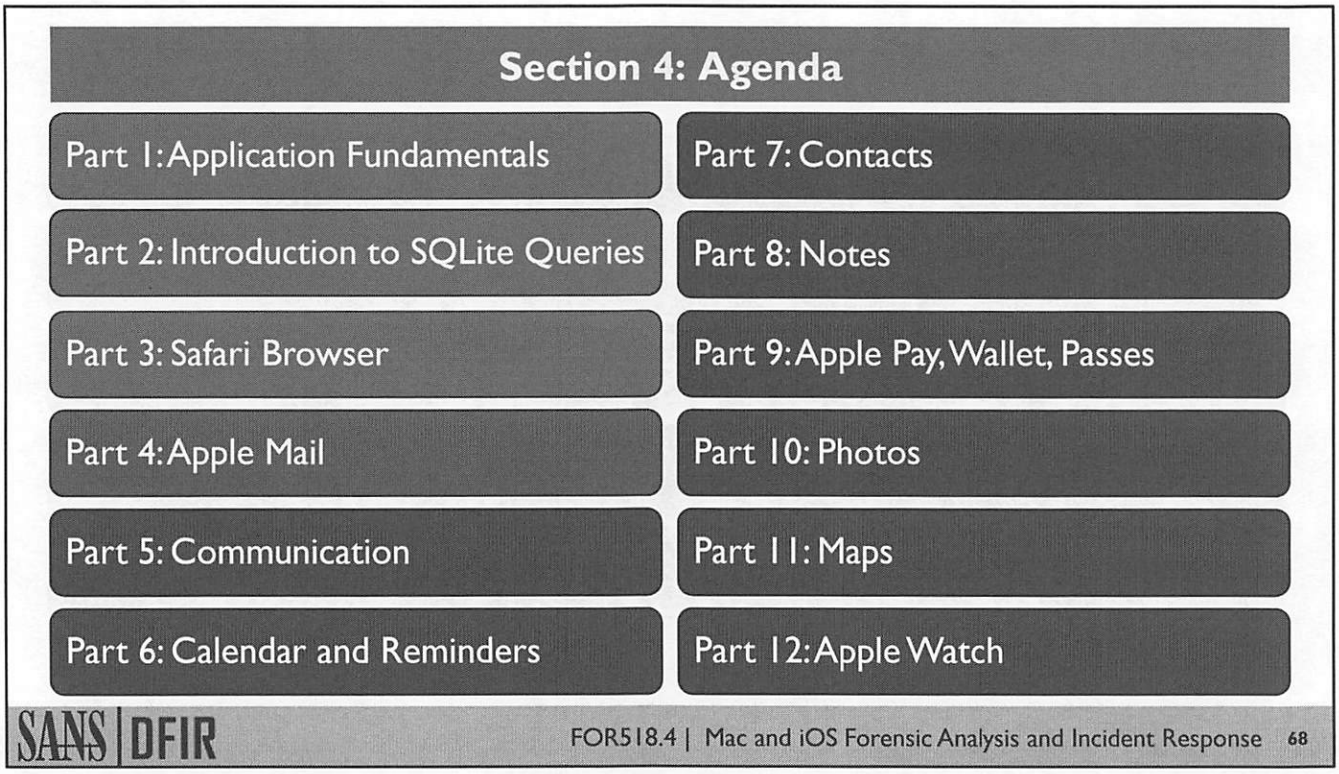


macOS: `~/Library/Cookies/`
iOS Backup: `/mobile/Library/Cookies/`
iOS Physical: `/private/var/mobile/Library/Cookies/`

On 10.6 and older systems, the cookies are stored in a plaintext XML property list file. On 10.7+ (iOS 6+) systems, the cookies are stored in the `Cookies.binarycookies` and `com.apple.Safari.SafeBrowsing.binarycookies`. This file starts with the file header "`cook`". The file format for these binary cookies is a proprietary format.

References:
Mari DeGrazia's absolutely fantastic Google Analytic Cookies Forensics presentation:
`https://github.com/mdegrazia/Presentations`
and her binary cookie parser, which is here: `https://github.com/mdegrazia/Safari-Binary-Cookie-Parser`

© 2020 Sarah Edwards



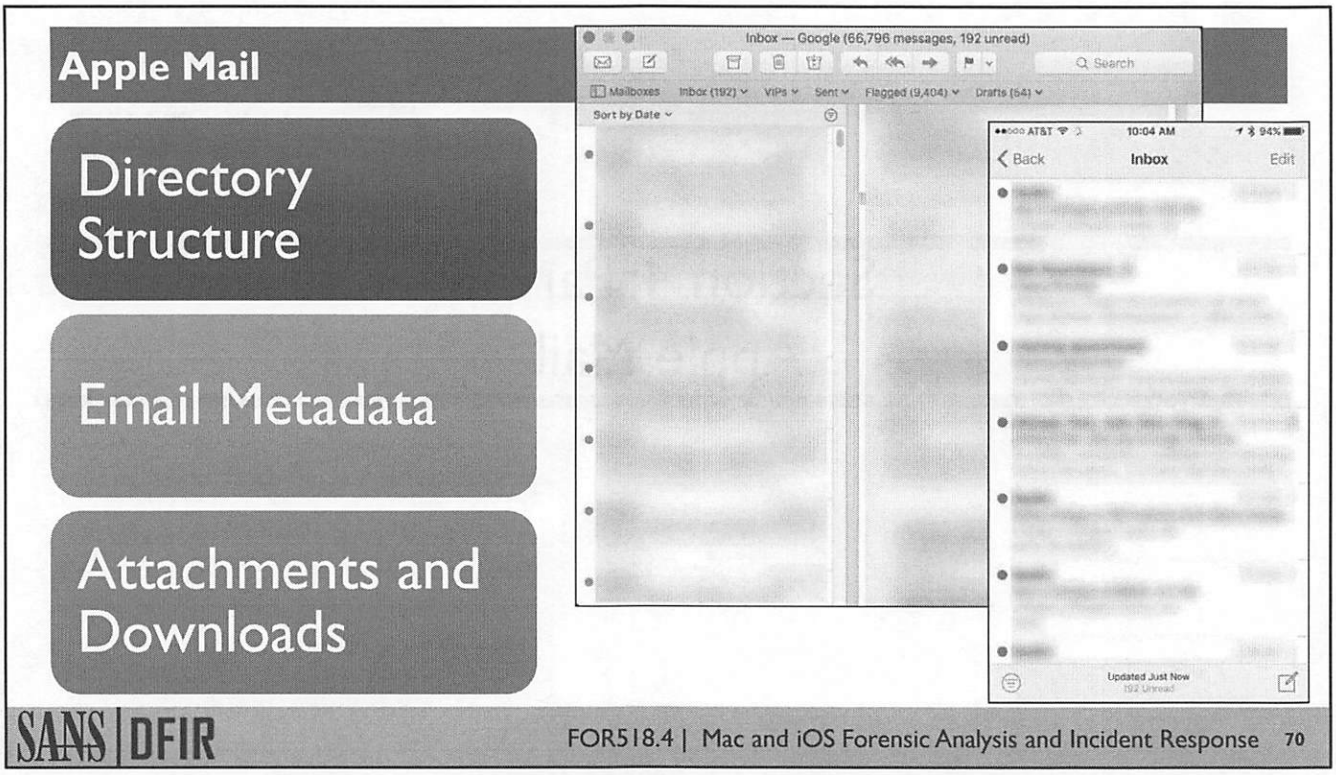
This page intentionally left blank.

© 2020 Sarah Edwards

Section 4: Part 4
Apple Mail

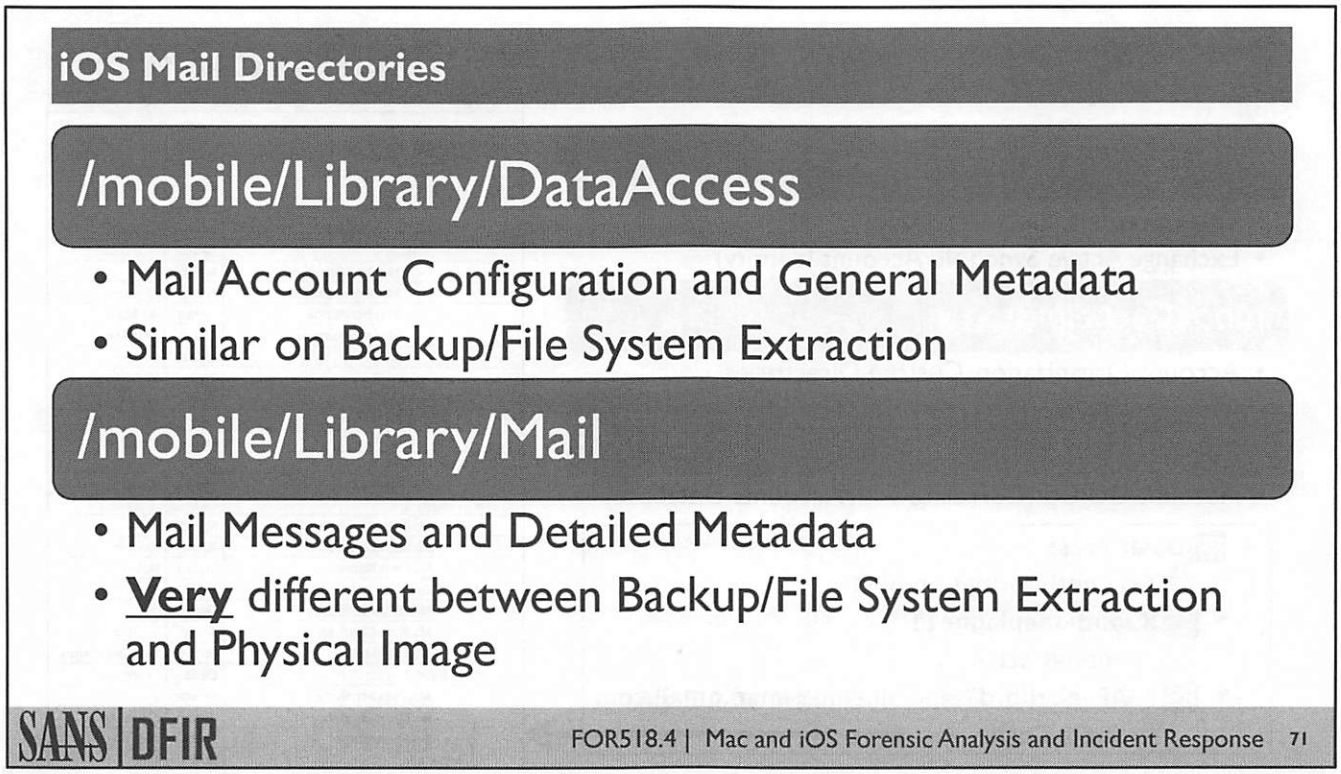
This page intentionally left blank.

© 2020 Sarah Edwards



The native email application on OS X is Apple Mail. Be sure to keep an eye out for other applications that may also be installed, such as Microsoft Outlook, Thunderbird, and even Lotus Notes!

© 2020 Sarah Edwards



The content of email is very different, depending on what type of acquisition you have. Each acquisition type has two areas where email data is found.

The `DataAccess` directory in /mobile/Library/ contains general email metadata and email account configuration information. This information is the same for all extractions.

The Mail directory in `/mobile/Library` is where things can be very different. Depending on the type of acquisition, you may have great information or you may have little to none.

© 2020 Sarah Edwards

iOS Mail: /mobile/Library/DataAccess/

Directories: Email Accounts

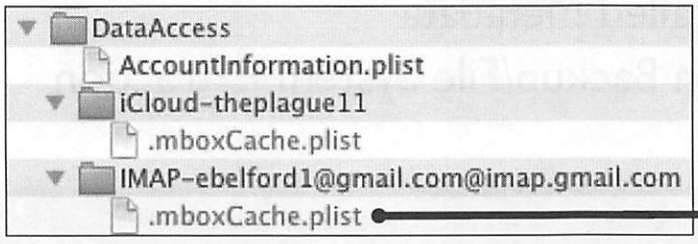
- Email Account Names/Protocol
- Exchange Active Sync (No Account Identity)

.mboxCache.plist

- Account Organization, Custom Directories

AccountInformation.plist

- Account GUIDS (Correlate with Accounts Database)



Key	Type	Value
capabilities	Array	(15 items)
mboxes	Array	(4 items)
Item 0	Dictio...	(3 items)
MailboxAttributes	Number	2
MailboxChildren	Array	(7 items)
Item 0	Dictio...	(4 items)
MailboxAttributes	Number	0
MailboxChildren	Array	(0 item)
MailboxName	String	All Mail
MailboxPermanentTag	String	\AllMail
Item 1	Dictio...	(4 items)
Item 2	Dictio...	(4 items)
Item 3	Dictio...	(4 items)
Item 4	Dictio...	(4 items)
Item 5	Dictio...	(4 items)
Item 6	Dictio...	(4 items)
MailboxName	String	[Gmail]
Item 1	Dictio...	(3 items)
MailboxAttributes	Number	0
MailboxChildren	Array	(0 item)
MailboxName	String	Notes
Item 2	Dictio...	(3 items)
MailboxAttributes	Number	0
MailboxChildren	Array	(0 item)
MailboxName	String	Sent Messages
Item 3	Dictio...	(3 items)
MailboxAttributes	Number	0
MailboxChildren	Array	(0 item)
MailboxName	String	INBOX
separator	String	/

72

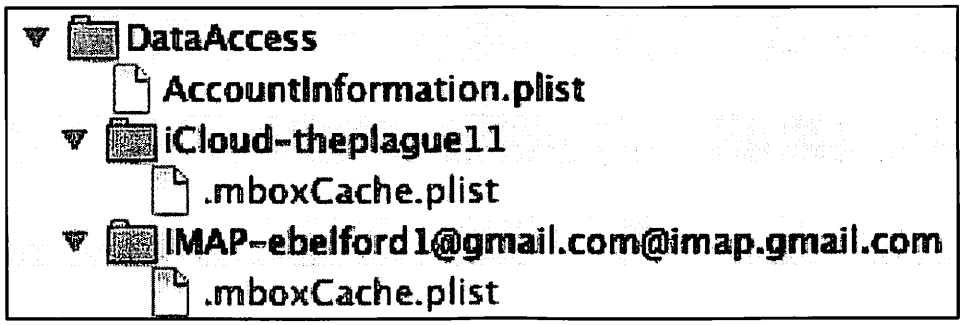
The `DataAccess` directory contains directories named after the email accounts they represent. For example, the `iCloud-thePlague11` is an iCloud email account (`theplague11@icloud.com`), while the other is a Gmail account (`ebelford1@gmail.com`).

The `.mboxCache.plist` files contain the organizational structure of each email account. This property list file contains the folder names the email messages are organized into.

In these files, you may find a folder of interest, depending on how the user has organized their email.

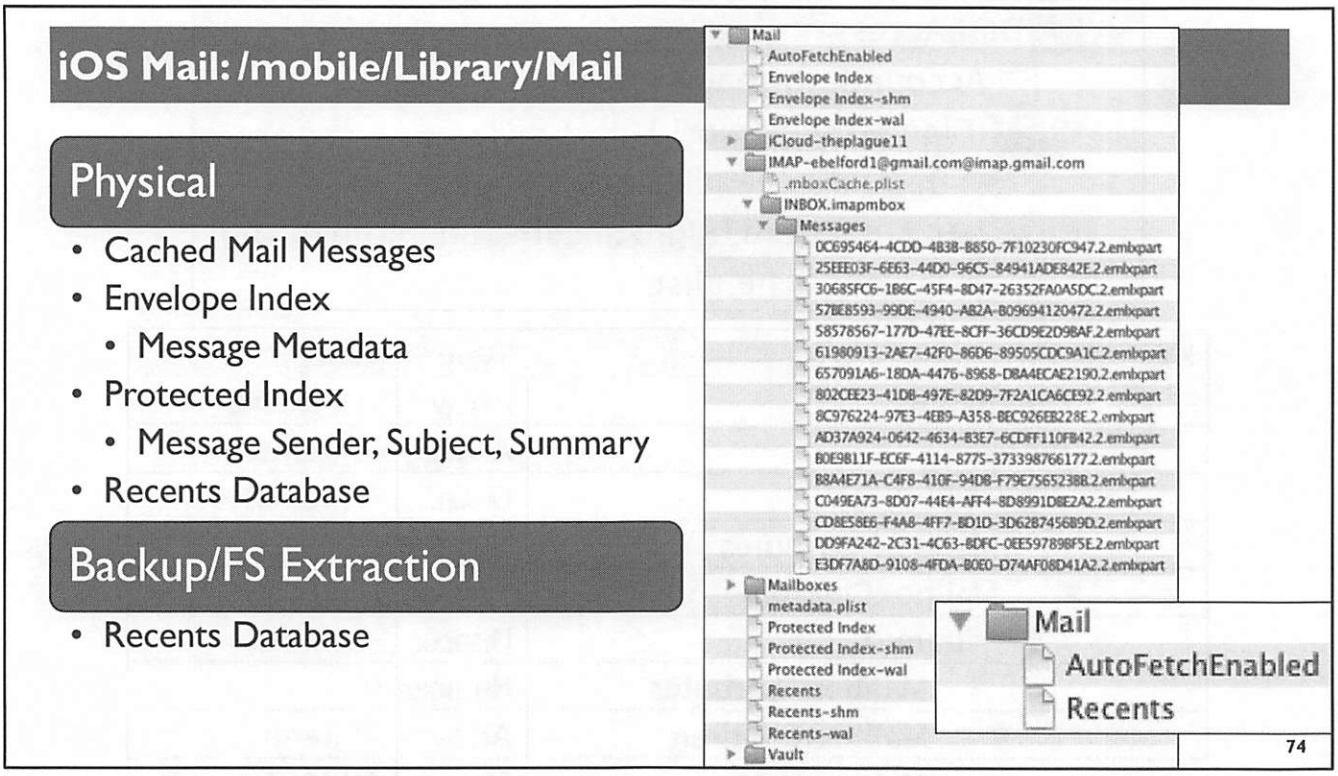
The `AccountInformation.plist` file contains the GUIDs that can be correlated with the Accounts database to determine additional account setup information.

© 2020 Sarah Edwards



Key	Type	Value
capabilities	Array	(15 items)
mboxes	Array	(4 items)
Item 0	Dictio...	(3 items)
MailboxAttributes	Number	2
MailboxChildren	Array	(7 items)
Item 0	Dictio...	(4 items)
MailboxAttributes	Number	0
MailboxChildren	Array	(0 item)
MailboxName	String	All Mail
MailboxPermanentTag	String	\AllMail
Item 1	Dictio...	(4 items)
Item 2	Dictio...	(4 items)
Item 3	Dictio...	(4 items)
Item 4	Dictio...	(4 items)
Item 5	Dictio...	(4 items)
Item 6	Dictio...	(4 items)
MailboxName	String	[Gmail]
Item 1	Dictio...	(3 items)
MailboxAttributes	Number	0
MailboxChildren	Array	(0 item)
MailboxName	String	Notes
Item 2	Dictio...	(3 items)
MailboxAttributes	Number	0
MailboxChildren	Array	(0 item)
MailboxName	String	Sent Messages
Item 3	Dictio...	(3 items)
MailboxAttributes	Number	0
MailboxChildren	Array	(0 item)
MailboxName	String	INBOX
separator	String	/

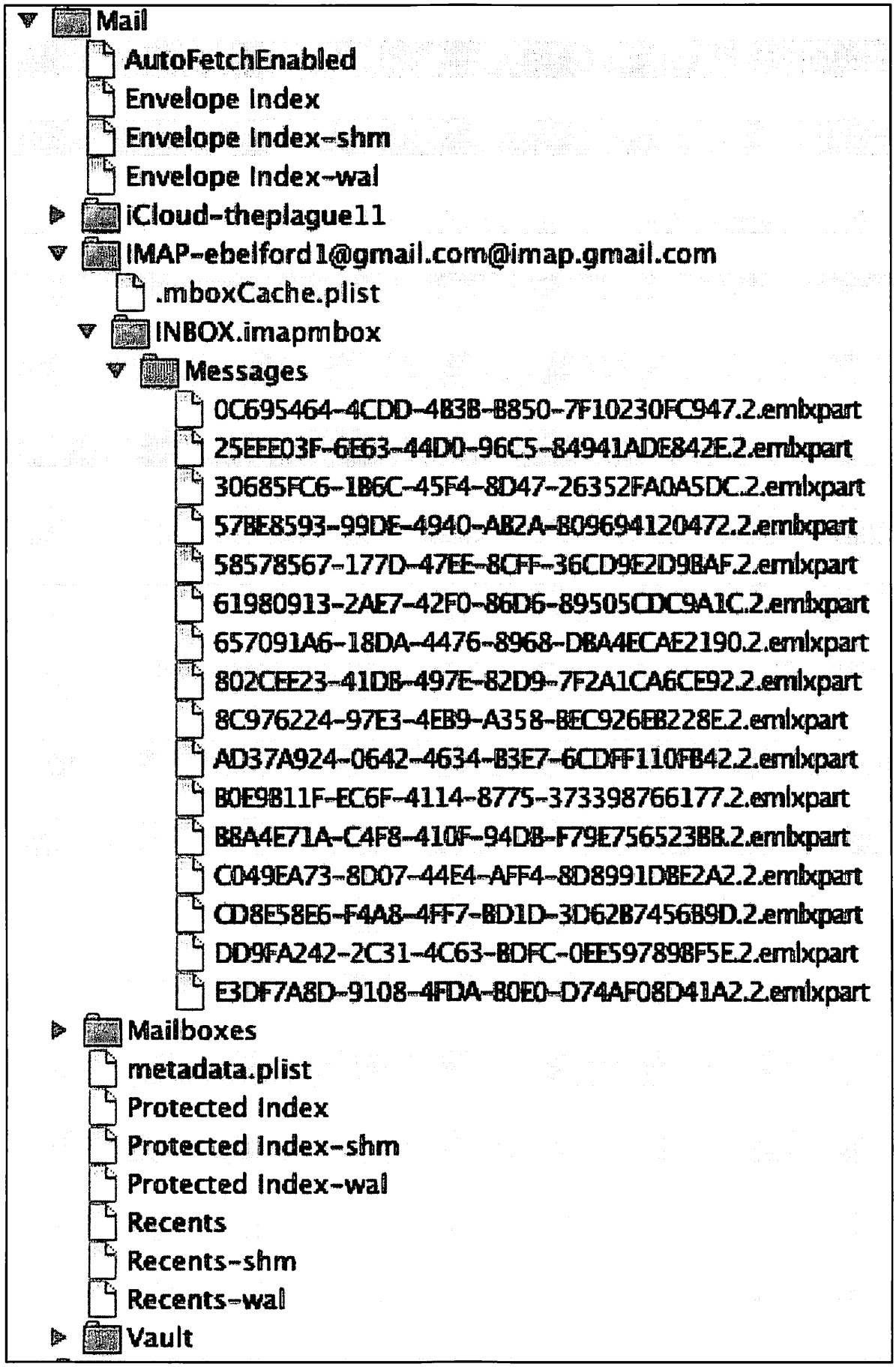
© 2020 Sarah Edwards



The `Mail` directory contains lots of information if you acquired a full physical (decrypted) image. Similar to an OS X system, you will have full email messages, the `Recents` database, and detailed message metadata in the `Envelope Index` databases.

If you only have a Backup or File System extraction, you will only get the `Recents` database, which contains information about recently messaged contacts, similar to the same database on OS X.

© 2020 Sarah Edwards



© 2020 Sarah Edwards

macOS Mail:V3/V4/V5 – ~/Library/Mail/V#/ [10.11 V3] [10.12 V4] [10.13 V5] [10.14 V6]

Directory for each email account

- GUID directory name
- Can be correlated in the `Accounts[3|4].sqlite` database

MailData: Mail Application Data

```
Sarahs-Air:V6 oompa$ pwd
/Users/oompa/Library/Mail/V6
Sarahs-Air:V6 oompa$ ls -la
total 0
drwxr-xr-x@  8 oompa  staff  256 Nov 29 19:37 .
drwx------@  4 oompa  staff  128 Nov 29 19:37 ..
drwxr-xr-x@ 10 oompa  staff  320 Jan  1 14:01 49285778-1207-4AB3-8F55-EE3A063B6743
drwxr-xr-x@  5 oompa  staff  160 Dec 31 22:02 A1A1F4DD-809F-48A5-A15B-6714993A3D63
drwxr-xr-x@ 10 oompa  staff  320 Jan  1 14:01 C59BB561-B88F-4D21-B709-9F72F84DFF80
drwxr-xr-x@ 19 oompa  staff  608 Jan  1 13:41 F37CEC82-FE8E-46B6-B236-CBF1D57D5D85
drwxr-xr-x@  4 oompa  staff  128 Jan  1 11:48 F8B67AC8-357E-477E-93C3-82B9FBE1BDBB
drwxr-xr-x@ 21 oompa  staff  672 Jan  1 13:55 MailData
```


The "version" number of the Mail directory has changed over the last few macOS versions:

- 10.11 = V3
- 10.12 = V4
- 10.13 = V5
- 10.14 = V6

In these directories, you will find GUID-named directories. These can be correlated with various email accounts by using the `Accounts3.sqlite` and `Accounts4.sqlite` database files.

Each of these GUID-based files contains a similar structure to that found in older versions, as we will see in the next few slides.

© 2020 Sarah Edwards

macOS Mail: Email Accounts : ~/Library/Mail/V#/[GUID]/

Each *.mbox file is a mailbox

An account may have multiple mailboxes

- Inbox
- Sent Messages
- [Gmail]
- Deleted Messages
- Notes
- Drafts
- User Created Mailboxes

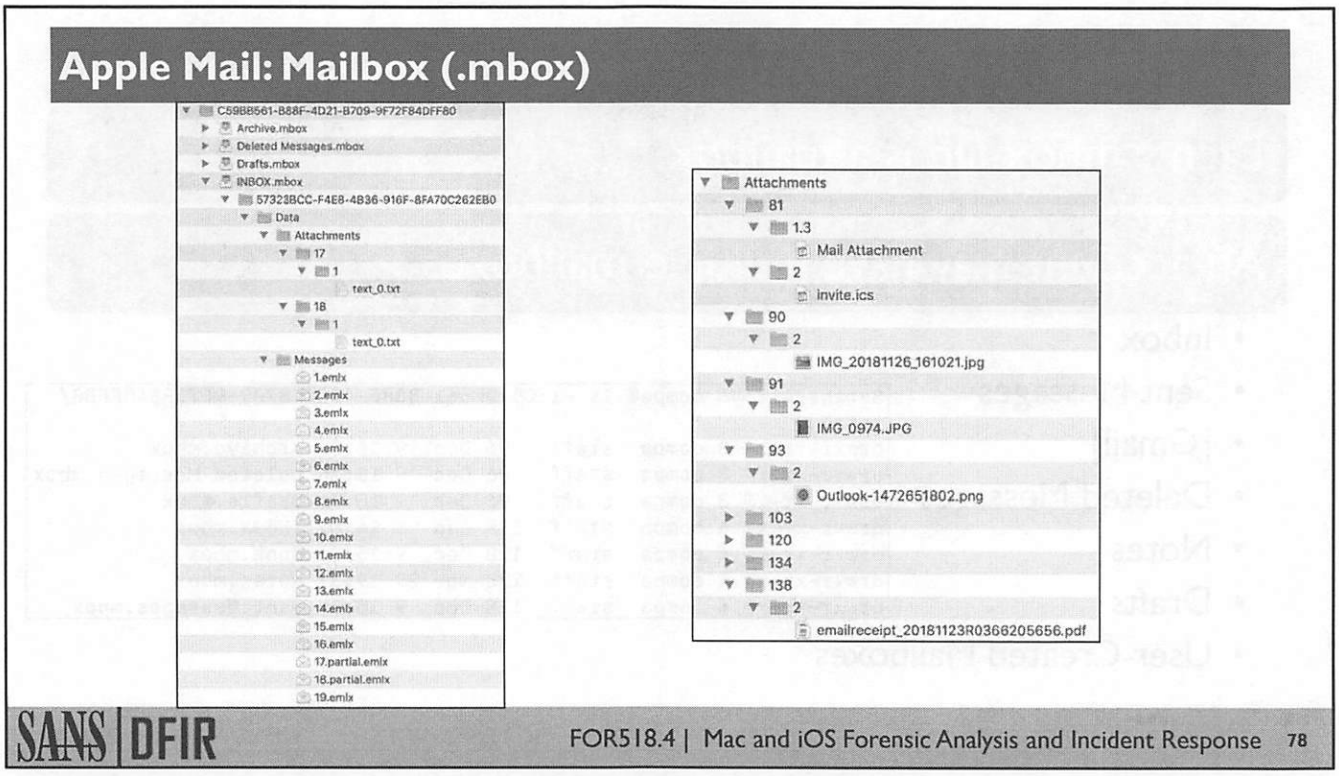
```
Sarahs-Air:V6 oompa$ ls -l C59BB561-B88F-4D21-B709-9F72F84DFF80/
total 0
drwxr-xr-x@ 3 oompa  staff   96 Dec  9 15:36 Archive.mbox
drwxr-xr-x@ 3 oompa  staff   96 Dec  9 15:36 Deleted Messages.mbox
drwxr-xr-x@ 3 oompa  staff   96 Dec  9 15:36 Drafts.mbox
drwxr-xr-x@ 4 oompa  staff  128 Jan  1 14:02 INBOX.mbox
drwxr-xr-x@ 4 oompa  staff  128 Dec  9 15:36 Junk.mbox
drwxr-xr-x@ 4 oompa  staff  128 Nov 29 19:37 Notes.mbox
drwxr-xr-x@ 4 oompa  staff  128 Dec  9 15:36 Sent Messages.mbox
```


Each Mailbox Account GUID directory contains one or more mailbox (`.mbox`) directories.

The screenshot shows three mailbox (`.mbox`) directories.

- `Archive.mbox`
- `Deleted Messages.mbox`
- `Drafts.mbox`
- `INBOX.mbox`
- `Junk.mbox`
- `Notes.mbox`
- `Sent Messages.mbox`

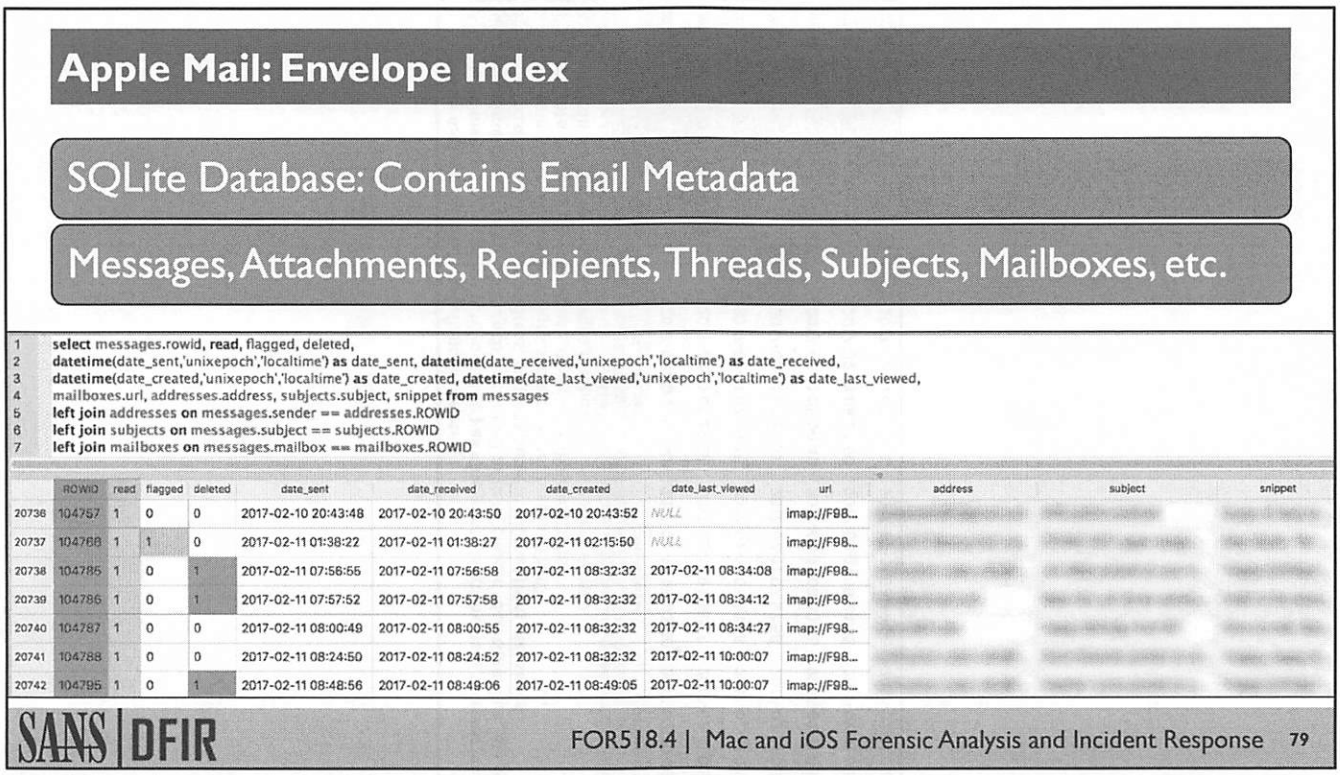
© 2020 Sarah Edwards



The GUID directory contains the raw email messages (`.emlx`) and email metadata in the Messages directory.

- Messages: Contains the raw email messages (`.emlx`), with an appended property list containing message metadata.
- Attachments: Contains the message file attachments.

© 2020 Sarah Edwards



The SQLite database `Envelope Index`, located in the `MailData` directory, contains indexed mail data.

- The `addresses` table contains all the indexed email addresses and associated contact names.
- The `attachments` table contains the name of each attachment.
- The `mailboxes` table contains data for each mailbox, including total messages and unread messages.
- The `messages` table contains metadata for each email message, including To, From, Subject, email timestamps, and if the message was read or not.
- The `subjects` table contains the email subject for each email message.

In the screenshot above, the actual email can be found by looking for the `ROWID`. The message will be named `104757.emlx`. Related attachments will also have the associated number.

Emails that have been read will contain a '`1`' in the `read` column, as will messages that have been `flagged` or `deleted`, in their respective columns. The `url` column shows the path to the email account—this will contain the GUID associated with the account (also found in the `Accounts` database). The address column shows the sender's email address.

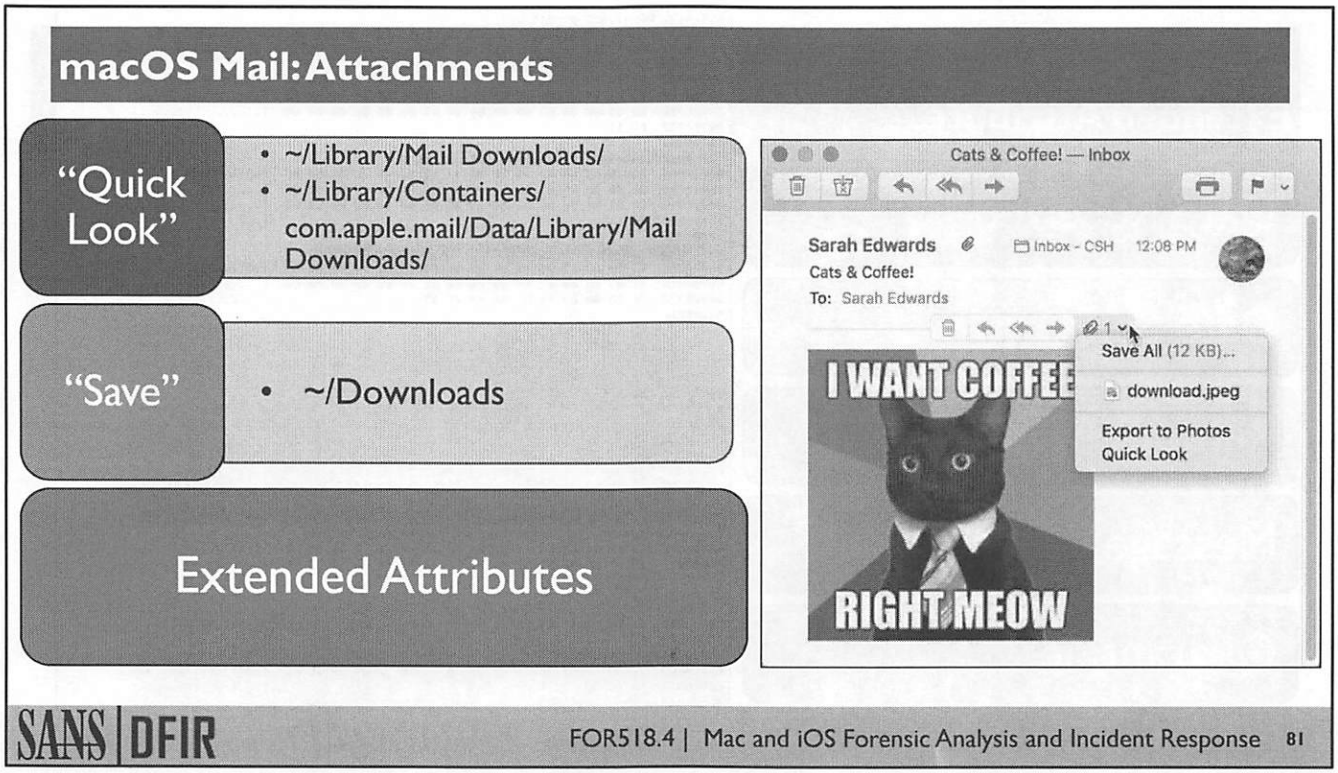
```
select messages.rowid, read, flagged, deleted,
datetime(date_sent,'unixepoch','localtime') as date_sent,
datetime(date_received,'unixepoch','localtime') as date_received,
datetime(date_created,'unixepoch','localtime') as date_created,
datetime(date_last_viewed,'unixepoch','localtime') as date_last_viewed,
mailboxes.url, addresses.address, subjects.subject, snippet from messages
left join addresses on messages.sender == addresses.ROWID
left join subjects on messages.subject == subjects.ROWID
left join mailboxes on messages.mailbox == mailboxes.ROWID
```

© 2020 Sarah Edwards

```
1  select messages.rowid, read, flagged, deleted,
2  datetime(date_sent,'unixepoch','localtime') as date_sent, datetime(date_received,'unixepoch','localtime') as date_received,
3  datetime(date_created,'unixepoch','localtime') as date_created, datetime(date_last_viewed,'unixepoch','localtime') as date_last_viewed,
4  mailboxes.url, addresses.address, subjects.subject, snippet from messages
5  left join addresses on messages.sender == addresses.ROWID
6  left join subjects on messages.subject == subjects.ROWID
7  left join mailboxes on messages.mailbox == mailboxes.ROWID
```

	ROWID	read	flagged	deleted	date_sent	date_received	date_created	date_last_viewed	url	address	subject	snippet
20736	104757	1	0	0	2017-02-10 20:43:48	2017-02-10 20:43:50	2017-02-10 20:43:52	NULL	imap://F98...			
20737	104766	1	1	0	2017-02-11 01:38:22	2017-02-11 01:38:27	2017-02-11 02:15:50	NULL	imap://F98...			
20738	104785	1	0	1	2017-02-11 07:56:55	2017-02-11 07:56:58	2017-02-11 08:32:32	2017-02-11 08:34:08	imap://F98...			
20739	104786	1	0	1	2017-02-11 07:57:52	2017-02-11 07:57:58	2017-02-11 08:32:32	2017-02-11 08:34:12	imap://F98...			
20740	104787	1	0	0	2017-02-11 08:00:49	2017-02-11 08:00:55	2017-02-11 08:32:32	2017-02-11 08:34:27	imap://F98...			
20741	104788	1	0	0	2017-02-11 08:24:50	2017-02-11 08:24:52	2017-02-11 08:32:32	2017-02-11 10:00:07	imap://F98...			
20742	104795	1	0	1	2017-02-11 08:48:56	2017-02-11 08:49:06	2017-02-11 08:49:05	2017-02-11 10:00:07	imap://F98...			

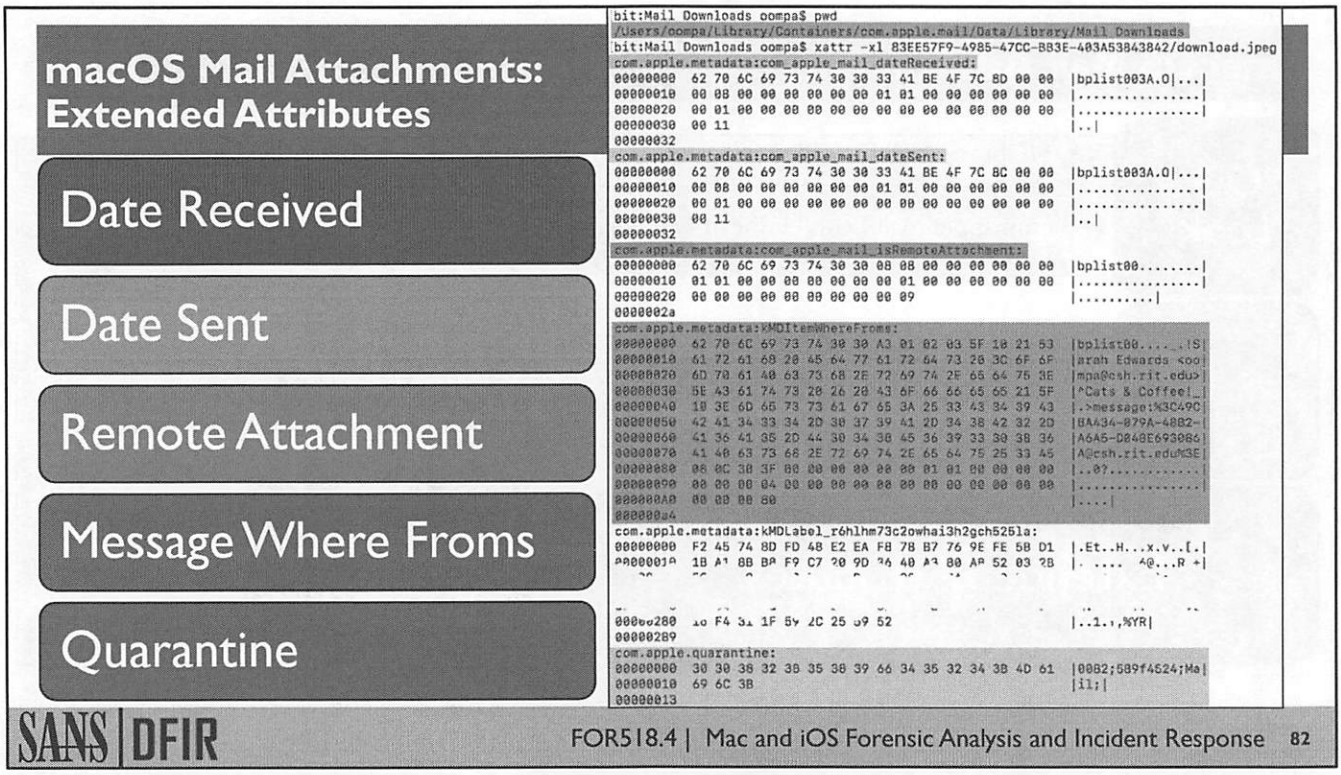
© 2020 Sarah Edwards



A user can view an attachment in a couple of ways:

- The "`Save`" button will save the attachment in the default Downloads directory, very likely `~/Downloads`.
- The "`Quick Look`" button opens the attachment for the viewer to see quickly and saves the attachment in the `~/Library/Mail Downloads/` directory. The sandbox directory may also be used `~/Library/Containers/com.apple.mail/Data/Library/Mail Downloads/`.

© 2020 Sarah Edwards

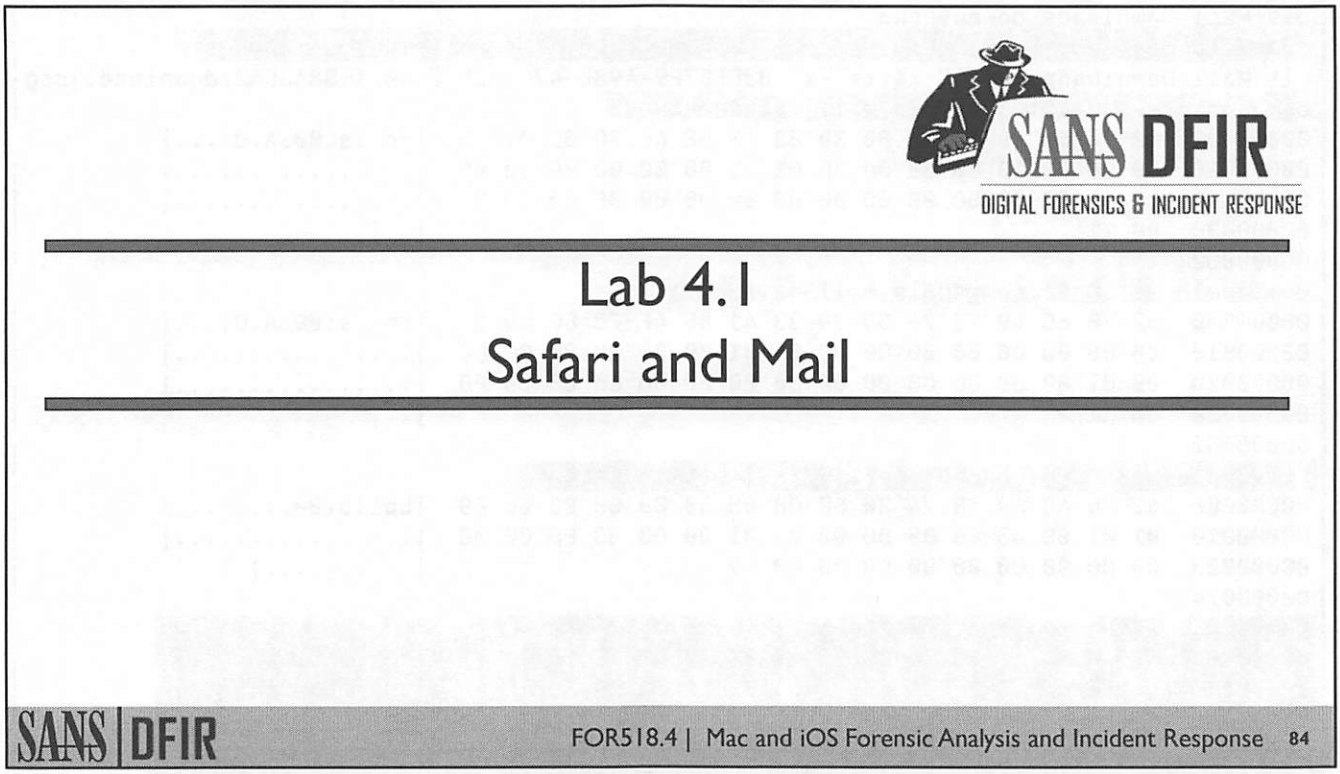


Each email attachment downloaded will have a set of metadata stored in its extended attributes. The example above shows the data from the previous slide's image attachment.

The Date Received, Sent, and whether it is a remote attachment are all stored in a binary plist. The email message "`Where Froms`" data shows the specific email message information, while the quarantine information shows where it came from (and when!).

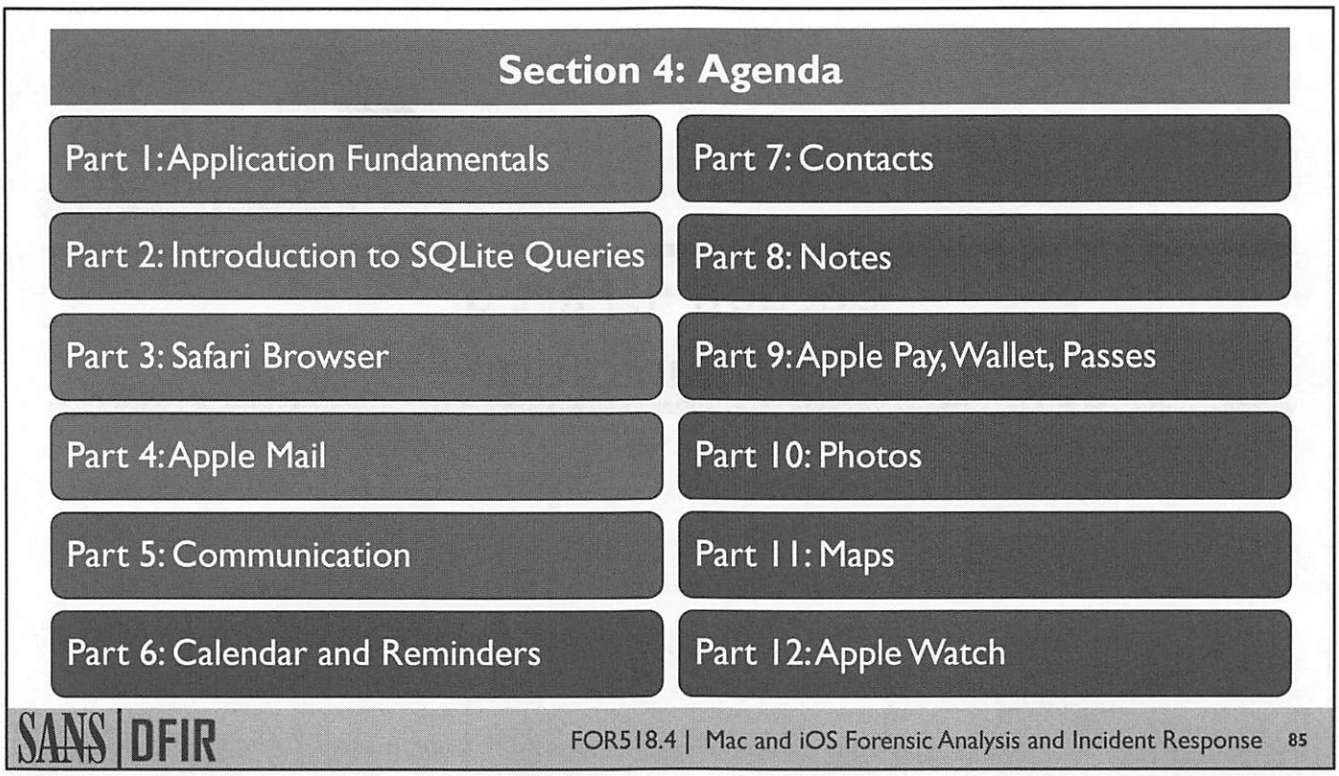
© 2020 Sarah Edwards

```
bit:Mail Downloads oompa$ pwd
/Users/oompa/Library/Containers/com.apple.mail/Data/Library/Mail Downloads
bit:Mail Downloads oompa$ xattr -xl 83EE57F9-4985-47CC-BB3E-403A53843842/download.jpeg
com.apple.metadata:com_apple_mail_dateReceived:
00000000  62 70 6C 69 73 74 30 30 33 41 BE 4F 7C 8D 00 00  |bplist003A.O|...|
00000010  00 08 00 00 00 00 00 00 01 01 00 00 00 00 00 00  |................|
00000020  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 11                                            |..|
00000032
com.apple.metadata:com_apple_mail_dateSent:
00000000  62 70 6C 69 73 74 30 30 33 41 BE 4F 7C 8C 00 00  |bplist003A.O|...|
00000010  00 08 00 00 00 00 00 00 01 01 00 00 00 00 00 00  |................|
00000020  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 11                                            |..|
00000032
com.apple.metadata:com_apple_mail_isRemoteAttachment:
00000000  62 70 6C 69 73 74 30 30 08 08 00 00 00 00 00 00  |bplist00........|
00000010  01 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00 00 09                    |..........|
0000002a
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A3 01 02 03 5F 10 21 53  |bplist00...._.!S|
00000010  61 72 61 68 20 45 64 77 61 72 64 73 20 3C 6F 6F  |arah Edwards <oo|
00000020  6D 70 61 40 63 73 68 2E 72 69 74 2E 65 64 75 3E  |mpa@csh.rit.edu>|
00000030  5E 43 61 74 73 20 26 20 43 6F 66 66 65 65 21 5F  |^Cats & Coffee!_|
00000040  10 3E 6D 65 73 73 61 67 65 3A 25 33 43 34 39 43  |.>message:%3C49C|
00000050  42 41 34 33 34 2D 30 37 39 41 2D 34 38 42 32 2D  |BA434-079A-48B2-|
00000060  41 36 41 35 2D 44 30 34 38 45 36 39 33 30 38 36  |A6A5-D048E693086|
00000070  41 40 63 73 68 2E 72 69 74 2E 65 64 75 25 33 45  |A@csh.rit.edu%3E|
00000080  08 0C 30 3F 00 00 00 00 00 00 01 01 00 00 00 00  |..0?............|
00000090  00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00  |................|
000000A0  00 00 00 80                                      |....|
000000a4
com.apple.metadata:kMDLabel_r6hlhm73c2owhai3h2gch525la:
00000000  F2 45 74 8D FD 48 E2 EA F8 78 B7 76 9E FE 5B D1  |.Et..H...x.v..[.|
00000010  1B A1 8B B8 F9 C7 20 9D 36 40 A3 80 AB 52 03 2B  |. .... 6@...R +|
[illegible]
00000280  16 F4 31 1F 59 2C 25 39 52                        |..1.Y,%YR|
00000289
com.apple.quarantine:
00000000  30 30 38 32 3B 35 38 39 66 34 35 32 34 3B 4D 61  |0082;589f4524;Ma|
00000010  69 6C 3B                                         |il;|
00000013
```



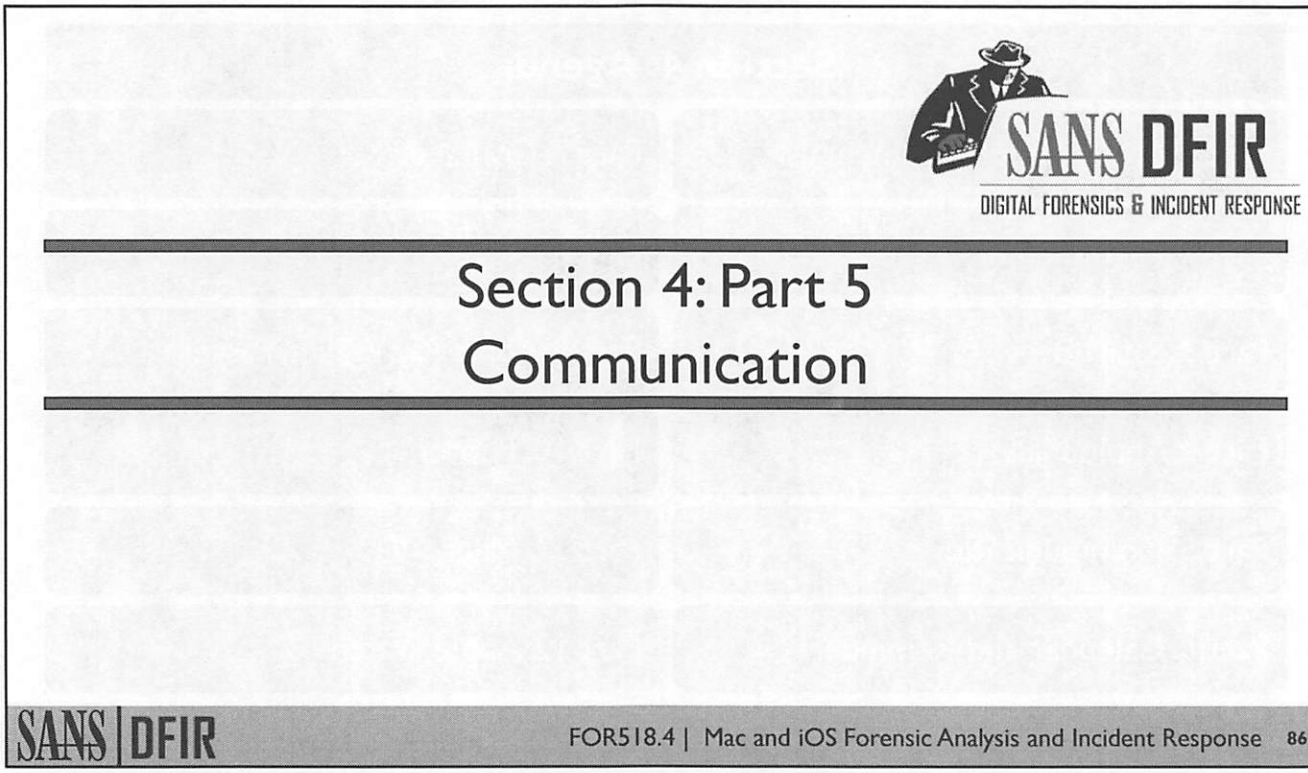
This page intentionally left blank.

© 2020 Sarah Edwards



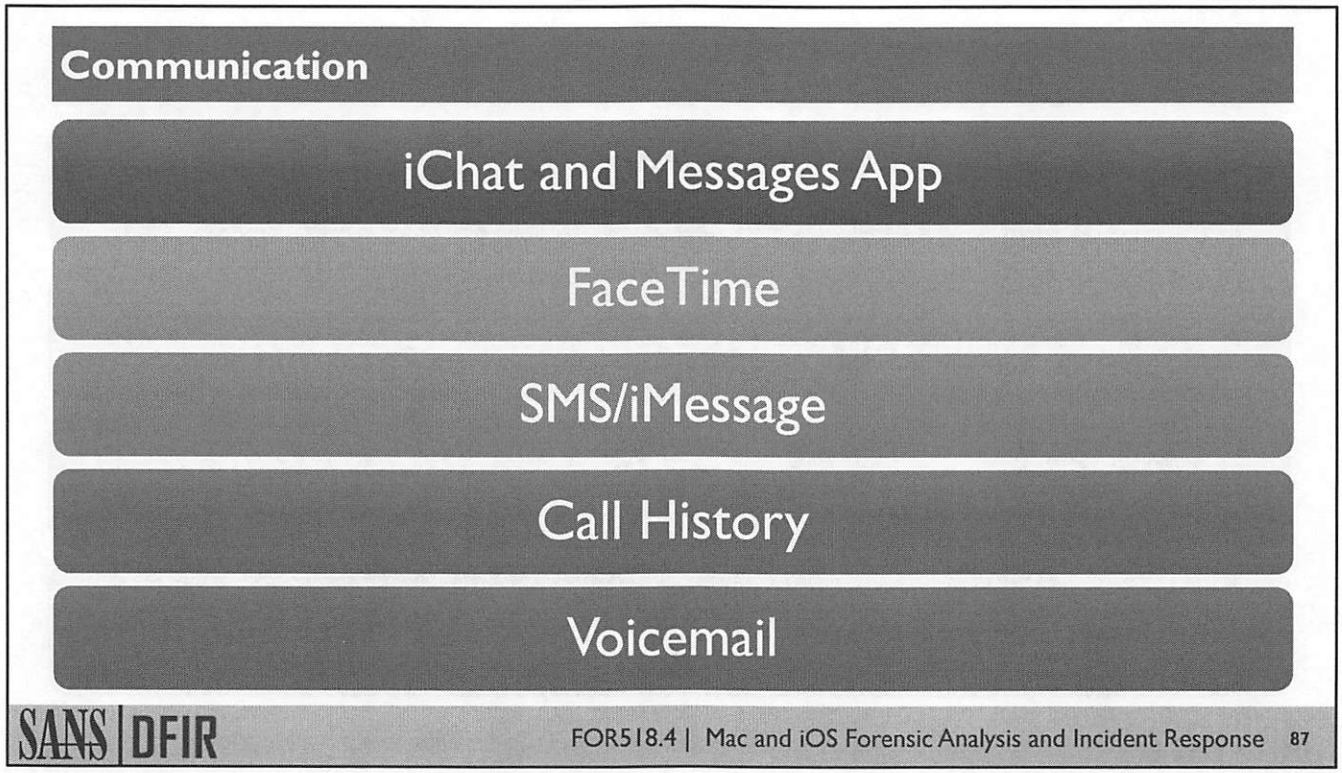
This page intentionally left blank.

© 2020 Sarah Edwards



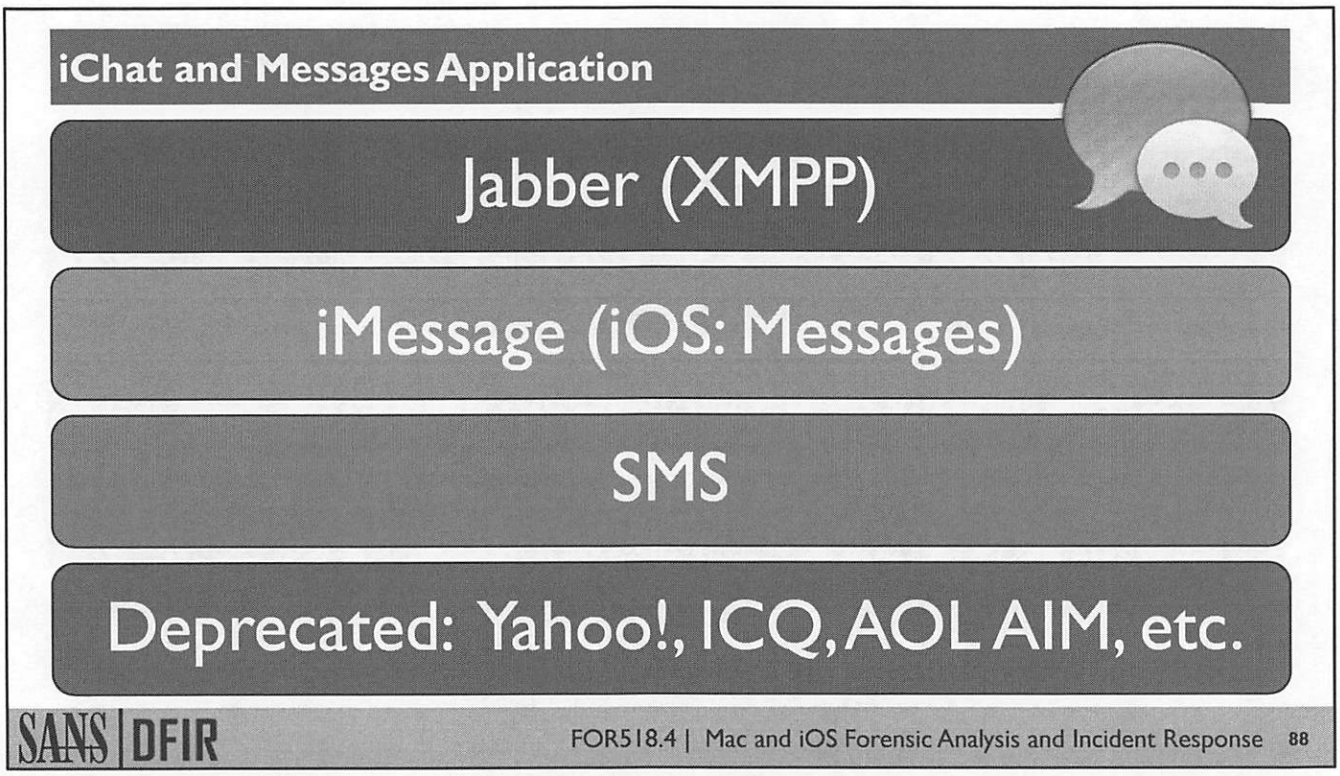
This page intentionally left blank.

© 2020 Sarah Edwards



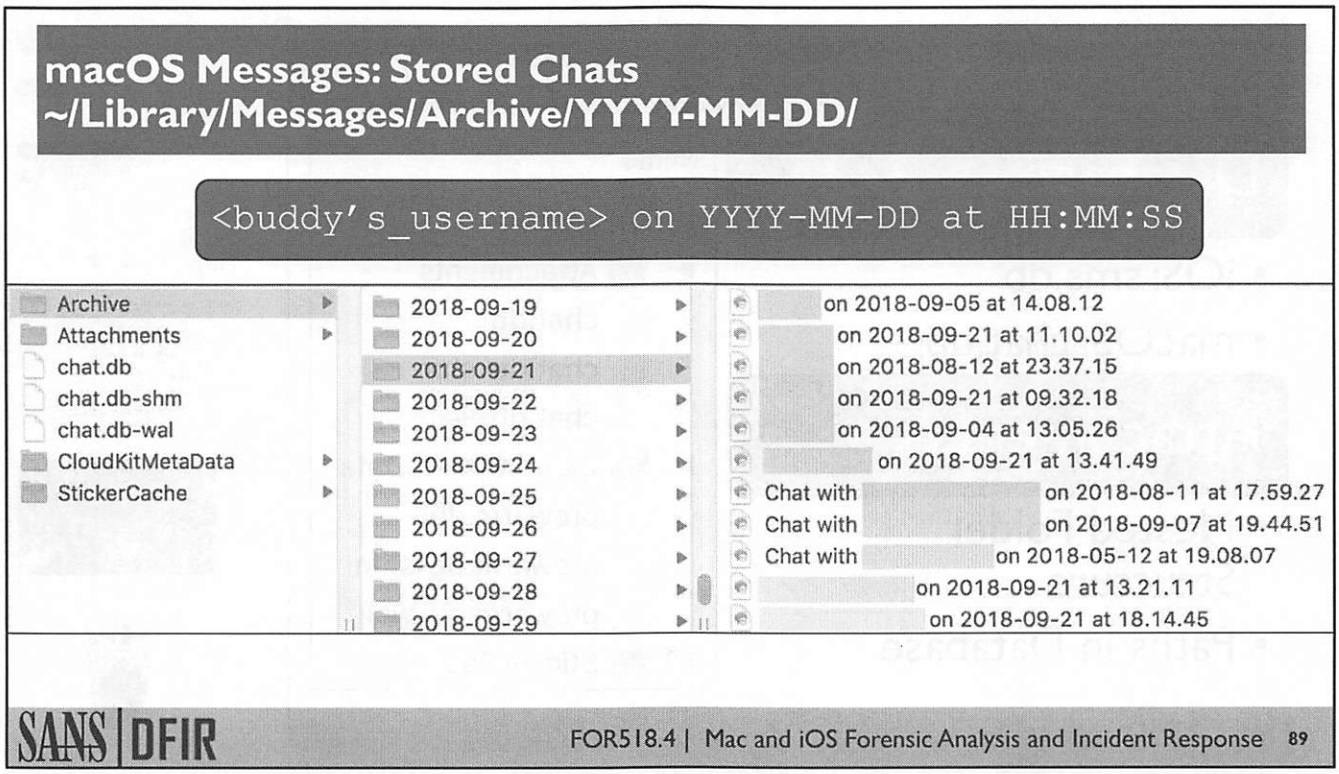
This page intentionally left blank.

© 2020 Sarah Edwards



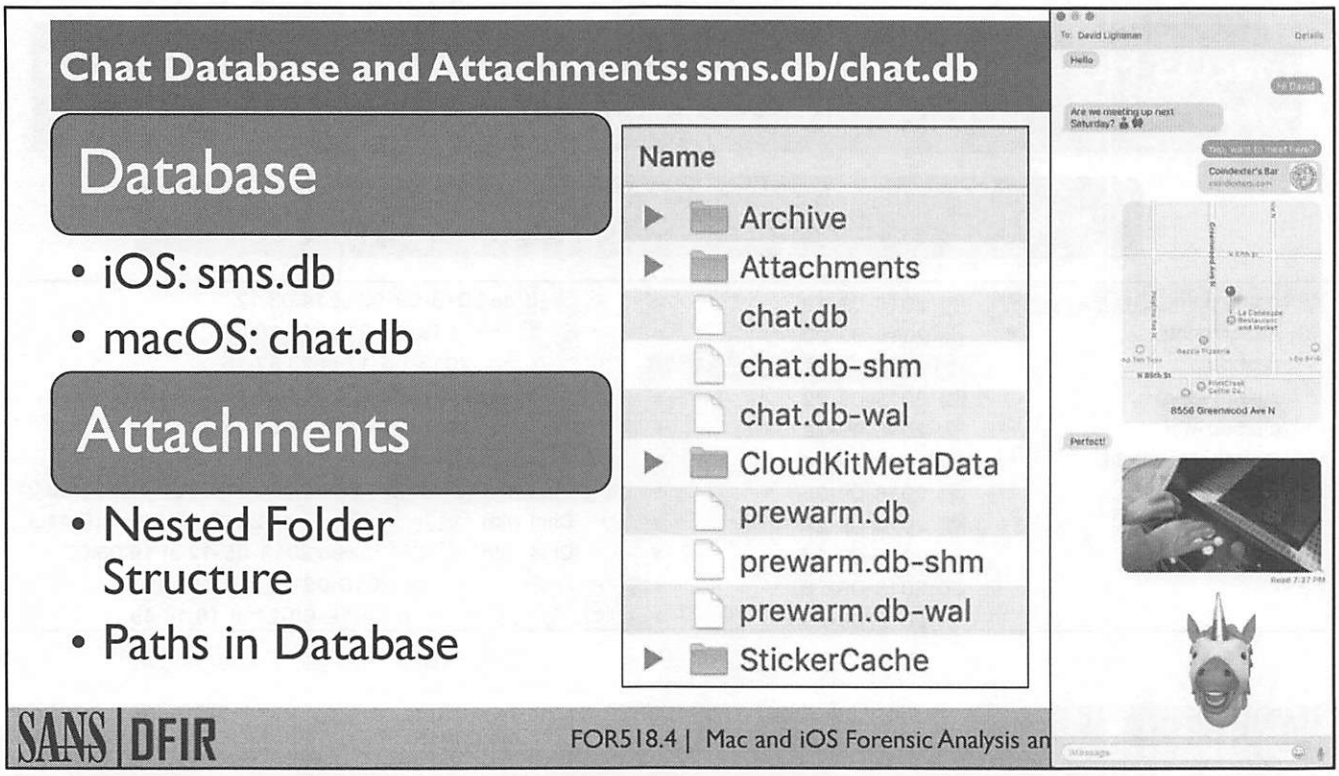
iChat (10.6 and 10.7) and its reincarnation, Messages (10.8+), are the native instant messaging applications of OS X. These applications can be used with a variety of instant messaging programs and protocols. Messages replaced iChat in 10.8 (Beta was available earlier) and introduced integration with iMessage (iOS) and FaceTime.

© 2020 Sarah Edwards



Messages will also store chat conversations if configured to do so. These files are stored in the `~/Library/Messages/Archive/` directory, also organized by date. These files use the same `iChat` binary property list format saved by date and time.

© 2020 Sarah Edwards



macOS: `~/Library/Messages/chat.db`
Physical: iOS 6+: `/private/var/mobile/Library/SMS/sms.db`
Backup/FS: iOS 6+: `/mobile/Library/SMS/sms.db`

The Messages application on iOS can be used for both iMessage and SMS-based messaging. The SMS directory will contain a SQLite database, `SMS.db`, and an `/Attachments` directory. On macOS the database is in the User's Messages directory and named chat.db.

The `SMS.db/chat.db` database file contains the message and related metadata, while the `/Attachments` directory contains media attachments.

© 2020 Sarah Edwards

Chat Database (sms.db/chat.db): Message Metadata

```
SELECT
CASE
    WHEN LENGTH(message.date)=18 THEN DATETIME(message.date/1000000000+978307200,'unixepoch','localtime')
    WHEN LENGTH(message.date)=9 THEN DATETIME(message.date +978307200,'unixepoch','localtime')
  else "N/A"
END AS "MESSAGE DATE",
CASE
    WHEN LENGTH(MESSAGE.DATE_DELIVERED)=18 THEN DATETIME(MESSAGE.DATE_DELIVERED/1000000000+978307200,'unixepoch','localtime')
    WHEN LENGTH(MESSAGE.DATE_DELIVERED)=9 THEN DATETIME(MESSAGE.DATE_DELIVERED +978307200,'unixepoch','localtime')
  else "N/A"
END AS "DATE DELIVERED",
CASE
    WHEN LENGTH(MESSAGE.DATE_READ)=18 THEN DATETIME(MESSAGE.DATE_READ/1000000000+978307200,'unixepoch','localtime')
    WHEN LENGTH(MESSAGE.DATE_READ)=9 THEN DATETIME(MESSAGE.DATE_READ +978307200,'unixepoch','localtime')
  else "N/A"
END AS "DATE READ",
MESSAGE.TEXT,HANDLE.ID AS "CONTACT ID", MESSAGE.SERVICE, MESSAGE.ACCOUNT, MESSAGE.IS_DELIVERED, MESSAGE.IS_FROM_ME, ATTACHMENT.FILENAME, ATTACHMENT.MIME_TYPE, ATTACHMENT.TRANSFER_NAME, ATTACHMENT.TOTAL_BYTES
FROM MESSAGE
LEFT OUTER JOIN MESSAGE_ATTACHMENT_JOIN ON MESSAGE.ROWID = MESSAGE_ATTACHMENT_JOIN.MESSAGE_ID
LEFT OUTER JOIN ATTACHMENT ON MESSAGE_ATTACHMENT_JOIN.ATTACHMENT_ID = ATTACHMENT.ROWID
LEFT OUTER JOIN HANDLE ON MESSAGE.HANDLE_ID = HANDLE.ROWID
```

	MESSAGE DATE	DATE DELIVERED	DATE READ	text	CONTACT ID	service	account	is_delivered	is_from_me	filename
1	2019-05-05 18:55:55	2019-05-05 19:22:39	2019-05-05 19:22:39	Hello	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	0	NULL
2	2019-05-05 19:26:05	2019-05-05 19:26:08	2019-05-05 19:26:08	Hi David	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	NULL
3	2019-05-05 19:27:33	N/A	N/A	Are we meeting up next Saturday?	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	0	NULL
4	2019-05-05 19:33:25	2019-05-05 19:35:00	2019-05-05 19:35:00	Yep, want to meet here?	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	NULL
5	2019-05-05 19:33:29	2019-05-05 19:35:00	2019-05-05 19:35:00	http://www.coindexters.com	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	~/Library/Messages/Attachments/76/06/BC1...
6	2019-05-05 19:34:47	2019-05-05 19:35:00	2019-05-05 19:35:00		d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	~/Library/Messages/Attachments/e1/01/D6BD...
7	2019-05-05 19:35:19	2019-05-05 19:35:39	2019-05-05 19:35:39	Perfect!	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	0	NULL
8	2019-05-05 19:37:16	2019-05-05 19:37:19	2019-05-05 19:37:19		d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	~/Library/Messages/Attachments/71/01/2BFF...
9	2019-05-05 19:41:10	N/A	N/A		d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	~/Library/Messages/Attachments/99/09/32B...

macOS: `~/Library/Messages/chat.db`
Physical: iOS 6+: `/private/var/mobile/Library/SMS/sms.db`
Backup/FS: iOS 6+: `/mobile/Library/SMS/sms.db`

A query on the `chat.db` SQLite database file shown below will extract the messages, timestamps, text, contact numbers/emails, originating account, attachment data, whether the message was delivered, and who originated it. The timestamps with "N/A" above originally contained a "0". These timestamps are in Mac Epoch. A specific CASE statement was created to deal with variable lengths of timestamps in the timestamp columns.

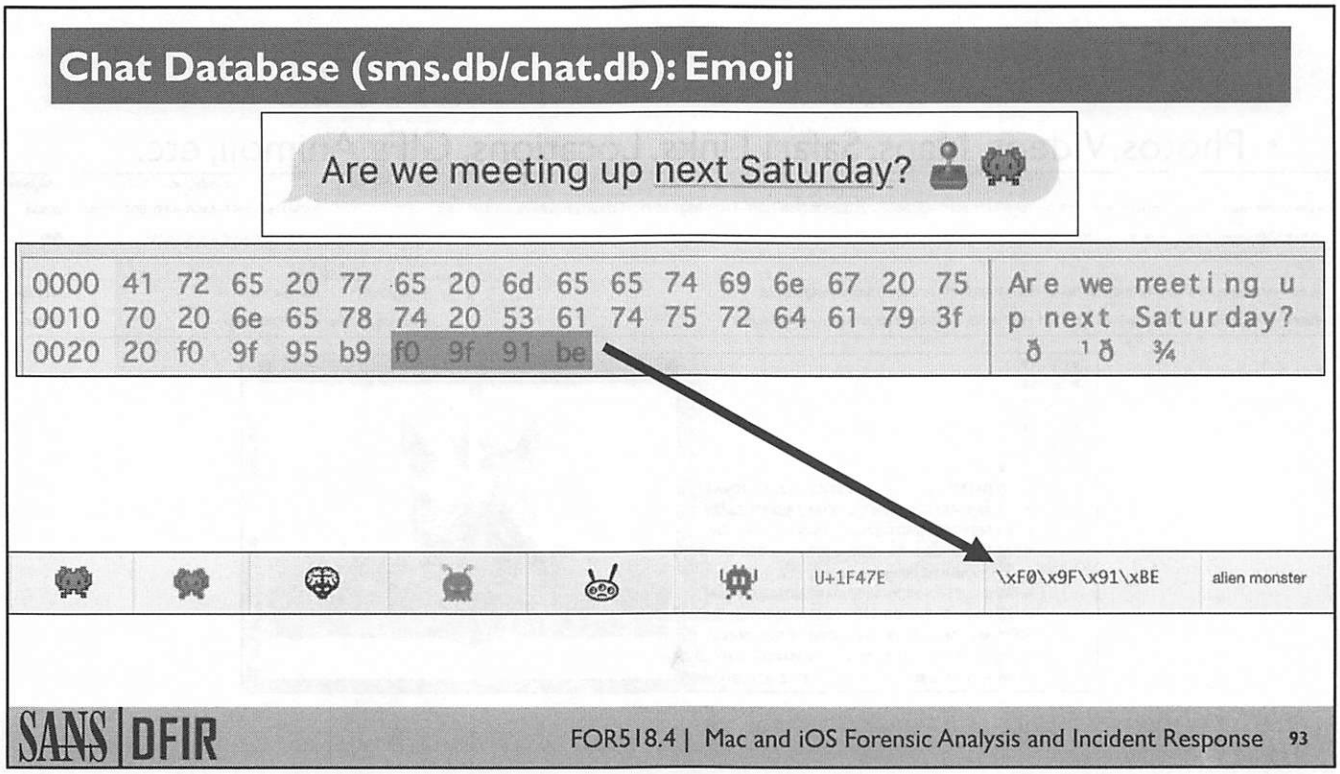
Reference:
`https://smarterforensics.com/2017/09/time-is-not-on-our-side-when-it-comes-to-messages-in-ios-11/`

© 2020 Sarah Edwards

```
SELECT
CASE
    WHEN LENGTH(message.date)=18 THEN DATETIME(message.date/1000000000+978307200,'unixepoch','localtime')
    WHEN LENGTH(message.date)=9 THEN DATETIME(message.date +978307200,'unixepoch','localtime')
 else "N/A"
END AS "MESSAGE DATE",
CASE
    WHEN LENGTH(MESSAGE.DATE_DELIVERED)=18 THEN DATETIME(MESSAGE.DATE_DELIVERED/1000000000+978307200,'unixepoch','localtime')
    WHEN LENGTH(MESSAGE.DATE_DELIVERED)=9 THEN DATETIME(MESSAGE.DATE_DELIVERED +978307200,'unixepoch','localtime')
 else "N/A"
END AS "DATE DELIVERED",
CASE
    WHEN LENGTH(MESSAGE.DATE_READ)=18 THEN DATETIME(MESSAGE.DATE_READ/1000000000+978307200,'unixepoch','localtime')
    WHEN LENGTH(MESSAGE.DATE_READ)=9 THEN DATETIME(MESSAGE.DATE_READ +978307200,'unixepoch','localtime')
 else "N/A"
END AS  "DATE READ",
MESSAGE.TEXT,HANDLE.ID AS "CONTACT ID", MESSAGE.SERVICE, MESSAGE.ACCOUNT, MESSAGE.IS_DELIVERED, MESSAGE.IS_FROM_ME, ATTACHMENT.FILENAME, ATTACHMENT.MIME_TYPE, ATTACHMENT.TRANSFER_NAME, ATTACHMENT.TOTAL_BYTES
FROM MESSAGE
LEFT OUTER JOIN MESSAGE_ATTACHMENT_JOIN ON MESSAGE.ROWID = MESSAGE_ATTACHMENT_JOIN.MESSAGE_ID
LEFT OUTER JOIN ATTACHMENT ON MESSAGE_ATTACHMENT_JOIN.ATTACHMENT_ID = ATTACHMENT.ROWID
LEFT OUTER JOIN HANDLE ON MESSAGE.HANDLE_ID = HANDLE.ROWID
```

	MESSAGE DATE	DATE DELIVERED	DATE READ	text	CONTACT ID	service	account	is_delivered	is_from_me	filename
1	2019-05-05 18:55:55	2019-05-05 19:22:39	2019-05-05 19:22:39	Hello	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	0	NULL
2	2019-05-05 19:26:05	2019-05-05 19:26:08	2019-05-05 19:26:08	Hi David	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	NULL
3	2019-05-05 19:27:33	N/A	N/A	Are we meeting up next Saturday?	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	0	NULL
4	2019-05-05 19:33:25	2019-05-05 19:35:00	2019-05-05 19:35:00	Yep, want to meet here?	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	NULL
5	2019-05-05 19:33:29	2019-05-05 19:35:00	2019-05-05 19:35:00	http://www.coindexters.com	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	~/Library/Messages/Attachments/76/06/BC1...
6	2019-05-05 19:34:47	2019-05-05 19:35:00	2019-05-05 19:35:00		d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	~/Library/Messages/Attachments/e1/01/D6BD...
7	2019-05-05 19:35:19	2019-05-05 19:35:39	2019-05-05 19:35:39	Perfect!	d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	0	NULL
8	2019-05-05 19:37:16	2019-05-05 19:37:19	2019-05-05 19:37:19		d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	~/Library/Messages/Attachments/71/01/2BFF...
9	2019-05-05 19:41:10	N/A	N/A		d.l1ghtm4n@gmail.com	iMessage	e:oompa@csh.rit.edu	1	1	~/Library/Messages/Attachments/99/09/32B...

© 2020 Sarah Edwards



macOS: `~/Library/Messages/chat.db`
Physical: iOS 6+: `/private/var/mobile/Library/SMS/sms.db`
Backup/FS: iOS 6+: `/mobile/Library/SMS/sms.db`

Messages may contain emojis. These messages can be reviewed from the binary view to see the Unicode that each emoji uses. This is handy if your SQLite viewer does not support Unicode.

The example above shows the message "`Are we meeting up next Saturday?`" with two emojis following. The highlighted emoji hex can be looked up in an emoji table like the one shown to determine which one it is.

References:
`https://apps.timwhitlock.info/emoji/tables/unicode`
`http://www.unicode.org/emoji/charts/full-emoji-list.html`

© 2020 Sarah Edwards

Chat Database (sms.db/chat.db): Chat Attachments

- Photos, Videos, Maps, Safari Links, Locations, GIFs, Animoji, etc.

filename	mime_type	transfer_name	total_bytes
~/Library/Messages/Attachments/76/06/8C1C3FDF-70F5-4611-89E0-46566E58F25D/88DB275B-DBFF-4ACB-9331-6D7101B97A90.pluginPayloadAttachment	NULL	88DB275B-DBFF-4ACB-9331-6D7101B97...	30534
~/Library/Messages/Attachments/e1/01/D6BD16C8-4757-4854-9088-9137C3199B12/8556 Greenwood Ave N.loc.vcf	text/x-vlocation	8556 Greenwood Ave N.loc.vcf	600
NULL	NULL	NULL	NULL
~/Library/Messages/Attachments/71/01/2BFF9457-0ED1-41A6-9FCB-DC64DD7F5A9A/ms-0ysL5s.gif	image/gif	ms-0ysL5s.gif	1007854
~/Library/Messages/Attachments/99/09/32B299C3-F0B2-4E03-9D03-D6CCBDC52424/ms-LAO9Ou.mov	video/quicktime	ms-LAO9Ou.mov	522190

macOS: `~/Library/Messages/chat.db`
Physical: iOS 6+: `/private/var/mobile/Library/SMS/sms.db`
Backup/FS: iOS 6+: `/mobile/Library/SMS/sms.db`

The "`filename`" column will hold a path to the attachment on the file system. An attachment does not have to be a picture or video. The "`mime_type`" column can suggest that other types of attachments are available, to include this list:

- "image/jpeg"
- "video/3gpp"
- "text/x-vlocation"
- "text/vcard"
- "image/png"
- "image/tiff"
- "video/quicktime"
- "audio/x-m4a"
- "image/gif"
- "audio/amr"

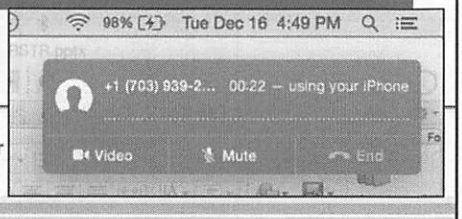
© 2020 Sarah Edwards

filename	mime_type	transfer_name	total_bytes
~/Library/Messages/Attachments/76/06/BC1C3FDF-70F5-4611-89E0-46566E58F25D/88DB275B-DBFF-4ACB-9331-6D7101B97A90.pluginPayloadAttachment	*NULL*	88DB275B-DBFF-4ACB-9331-6D7101B97...	30534
~/Library/Messages/Attachments/e1/01/D6BD16C8-4757-4854-9088-9137C3199B12/8556 Greenwood Ave N.loc.vcf	text/x-vlocation	8556 Greenwood Ave N.loc.vcf	600
NULL	*NULL*	*NULL*	*NULL*
~/Library/Messages/Attachments/71/01/2BFF9457-0ED1-41A6-9FCB-DC64DD7F5A9A/ms-0ysL5s.gif	image/gif	ms-0ysL5s.gif	1007854
~/Library/Messages/Attachments/99/09/32B299C3-F0B2-4E03-9D03-D6CCBDC52424/ms-LAO9Ou.mov	video/quicktime	ms-LAO9Ou.mov	522190

© 2020 Sarah Edwards

Newer Call Records: CallHistory.storedata [macOS10.10+, iOS 8+]

Call Records on macOS/iOS may be synced with macOS 10.12!




```
1  select datetime(zdate+978307200,'unixepoch','localtime') as zdate,
2  zaddress, zanswered, zcalltype, zoriginated, zduration, zlocation, zservice_provider
3  from ZCALLRECORD order by zdate desc
```

	zdate	ZADDRESS	ZANSWERED	ZCALLTYPE	ZORIGINATED	ZDURATION	ZLOCATION	ZSERVICE_PROVIDER
1	2017-02-09 14:33:14	+1	0	1	0	0.0	Virginia Beach, VA	com.apple.Telephony
2	2017-02-09 14:32:52	+1	0	1	0	0.0	Virginia Beach, VA	com.apple.Telephony
3	2017-02-07 18:09:10	+1	1	1	0	21.0	United States	com.apple.Telephony
4	2017-02-07 17:43:21	+1	1	1	0	42.0	United States	com.apple.Telephony
5	2017-02-07 17:42:17	+1	0	1	1	8.0	United States	com.apple.Telephony
6	2017-01-31 09:52:17	75	0	1	0	0.0	Portsmouth, VA	com.apple.Telephony
7	2017-01-30 20:09:42	+1	1	1	0	12.0	Arlington, VA	com.apple.Telephony
8	2017-01-26 19:36:00	+1	1	1	0	30.0	Arlington, VA	com.apple.Telephony

SANS | DFIR FOR518.4 | Mac and iOS Forensic Analysis and Incident Response 96

macOS: `~/Library/Application Support/CallHistoryDB/CallHistory.storedata`
iOS Physical: `/private/var/mobile/Library/CallHistoryDB/CallHistory.storedata`
iOS Backup: `/mobile/Library/CallHistoryDB/CallHistory.storedata`

The SQLite database `CallHistory.storedata` located in the directory contains the calls made. The screenshot in the upper right-hand corner contains the notification that the user will see when taking a call using their OS X system. This call data is then written to the database.

Each record contains the following data:

- `ZDATE`: Mac Epoch Timestamp
- `ZADDRESS`: Contact phone number or email address (FaceTime)
- `ZANSWERED`: 0 = No, 1 = Yes
- `ZCALLTYPE`: (If duration is 0, call was missed/canceled)
  - 1: Call
  - 8: FaceTime
  - 16: FaceTime Voice Call
- `ZORIGINATED`: 1 = Outgoing with this user, 0 = Incoming
- `ZDURATION`: Time in seconds of the call
- `ZLOCATION`: Location (from Mobile Records)
- `ZSERVICE_PROVIDER`: Application (`com.apple.Telephony`—Regular Call, may also see `com.apple.FaceTime`)

10.10–10.11: Only calls that were taken with the laptop or desktop system were recorded in this database. As of 10.12, the entire iPhone call history may be synced and can be accessed within the FaceTime application on macOS.

© 2020 Sarah Edwards

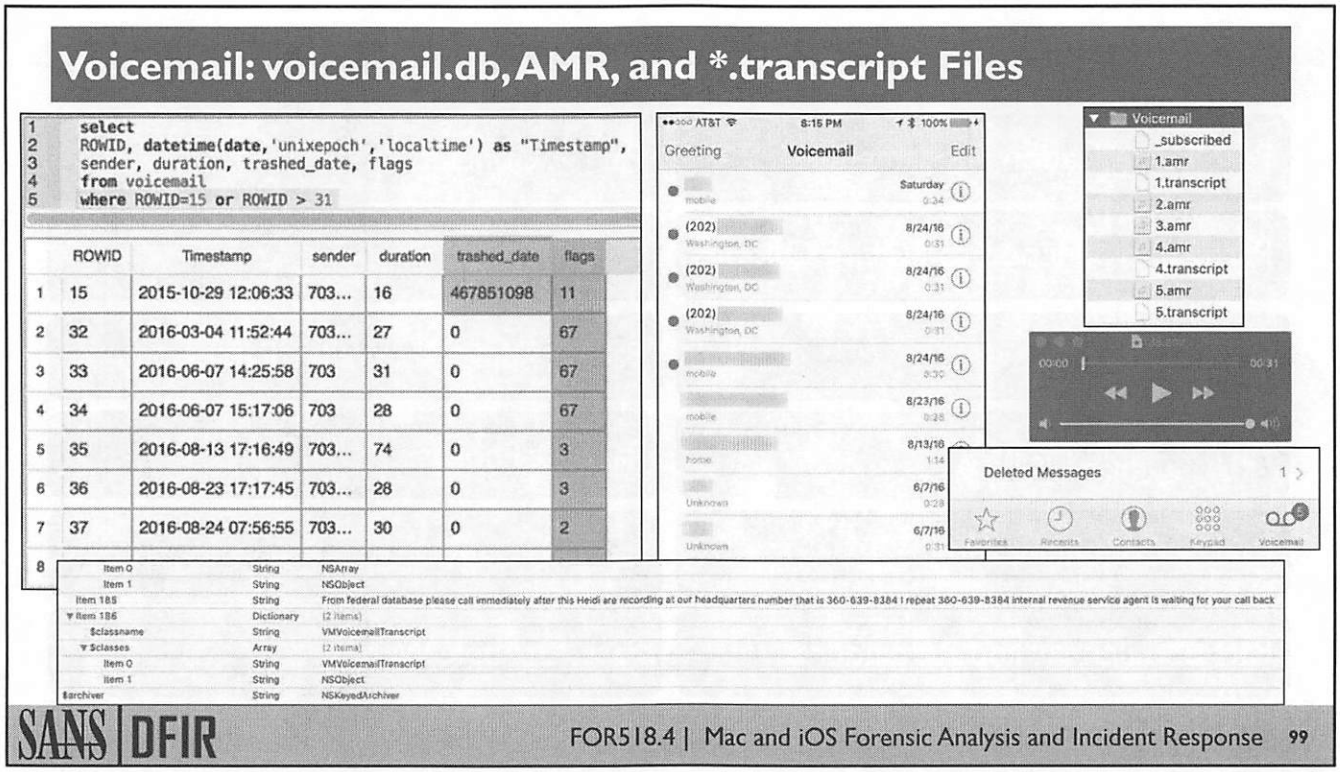
```
select
datetime(zdate+978307200,'unixepoch','localtime') as zdate,
zaddress,
zanswered,
zcalltype,
zoriginated,
zduration,
zlocation,
zservice_provider
from ZCALLRECORD
order by zdate desc
```

© 2020 Sarah Edwards

```
1  select datetime(zdate+978307200,'unixepoch','localtime') as zdate,
2  zaddress, zanswered, zcalltype, zoriginated, zduration, zlocation, zservice_provider
3  from ZCALLRECORD order by zdate desc
```

	zdate	ZADDRESS	ZANSWERED	ZCALLTYPE	ZORIGINATED	ZDURATION	ZLOCATION	ZSERVICE_PROVIDER
1	2017-02-09 14:33:14	+1	0	1	0	0.0	Virginia Beach, VA	com.apple.Telephony
2	2017-02-09 14:32:52	+1	0	1	0	0.0	Virginia Beach, VA	com.apple.Telephony
3	2017-02-07 18:09:10	+1	1	1	0	21.0	United States	com.apple.Telephony
4	2017-02-07 17:43:21	+1	1	1	0	42.0	United States	com.apple.Telephony
5	2017-02-07 17:42:17	+1	0	1	1	8.0	United States	com.apple.Telephony
6	2017-01-31 09:52:17	75	0	1	0	0.0	Portsmouth, VA	com.apple.Telephony
7	2017-01-30 20:09:42	+1	1	1	0	12.0	Arlington, VA	com.apple.Telephony
8	2017-01-26 19:36:00	+1	1	1	0	30.0	Arlington, VA	com.apple.Telephony

© 2020 Sarah Edwards



Physical: iOS 6+: `/private/var/mobile/Library/Voicemail/`
Backup/FS: iOS 6+: `/mobile/Library/Voicemail/`

If the iOS device has the visual voicemail functionality—not all cellular carriers implement this—the device will download the voicemail audio files to the phone. The Voicemail directory will contain a voicemail.db SQLite database file and downloaded AMR files.

The database query performed extracts the voicemail ID, timestamp, sender info, duration, if and when it was trashed, and flag information. If the voicemail is set to be deleted but is still on the device, it will show a timestamp of that date in Mac Epoch. The flags determine if the voicemail is deleted, downloaded, or unheard.

- 11: Trashed (may still be downloaded)
- 3/67: Downloaded Voicemail
- 2: Unheard

Each voicemail that is downloaded can be listened to using most audio players that support the Adaptive Multi-rate Audio (AMR) audio format. The voicemail filename will use the "`ROWID`" (voicemail ID) as its filename.

If the voicemail has an accompanying *.transcript file, you'll find an NSKeyedArchiver plist file with the transcript information.

© 2020 Sarah Edwards

Section 4: Agenda

- Part 1: Application Fundamentals
- Part 2: Introduction to SQLite Queries
- Part 3: Safari Browser
- Part 4: Apple Mail
- Part 5: Communication
- Part 6: Calendar and Reminders
- Part 7: Contacts
- Part 8: Notes
- Part 9: Apple Pay, Wallet, Passes
- Part 10: Photos
- Part 11: Maps
- Part 12: Apple Watch

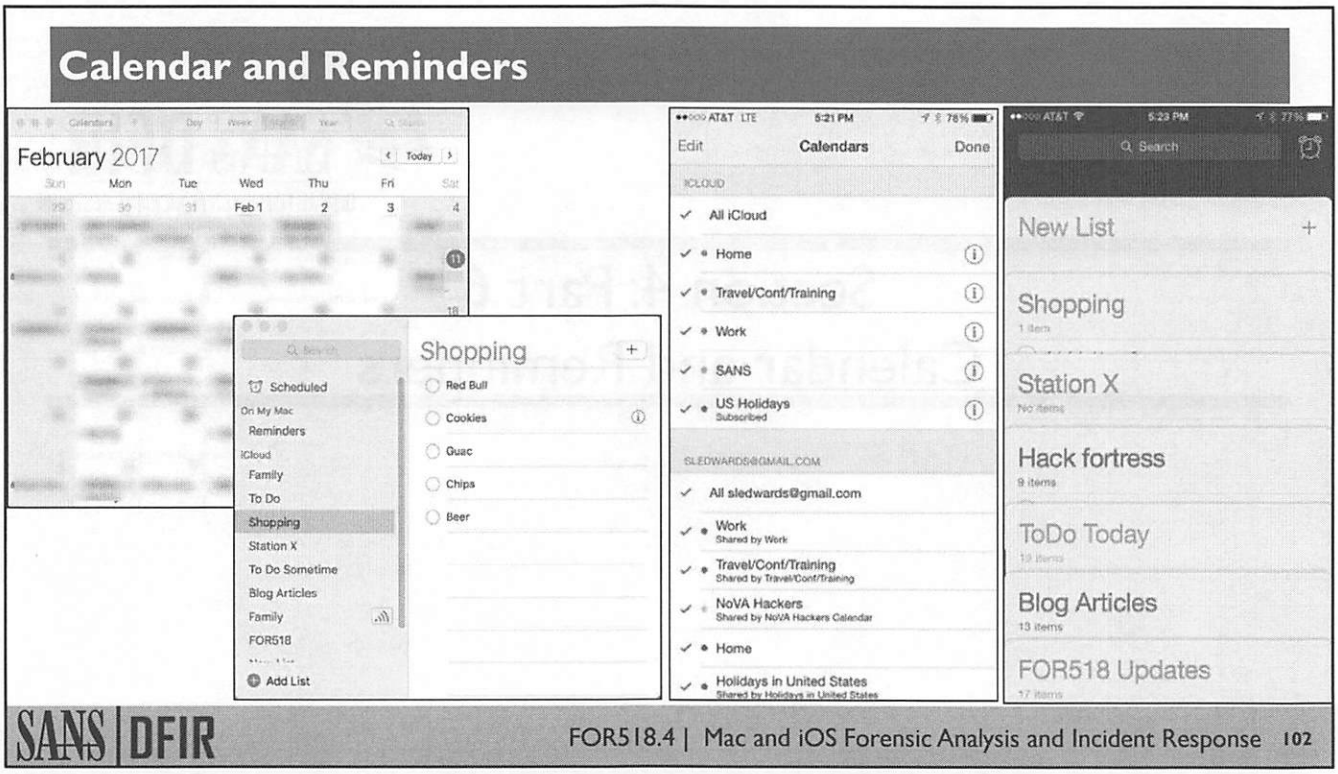
This page intentionally left blank.

© 2020 Sarah Edwards

Section 4: Part 6
Calendar and Reminders

This page intentionally left blank.

© 2020 Sarah Edwards



The native calendar application is iCal or Calendar. The name changed to Calendar in Mountain Lion (10.8). The Reminders application was added later.

The Calendar and Reminders applications use the same database to store their contents.

These items can be synced from a variety of accounts, including iCloud and Google, as shown above. It can include both personal calendars and shared calendars.

The Reminders application stores lists created by the user that may have deadlines and other information associated with each reminder item.

© 2020 Sarah Edwards

macOS: iCal / Calendar—Calendar Directories

```
Elwoods-Mac:Calendars elwoodblues$ pwd
/Users/elwoodblues/Library/Calendars
Elwoods-Mac:Calendars elwoodblues$ ls -l
total 424
drwxr-xr-x  4 elwoodblues  staff     136 Sep 23 11:28 788EEBAA-B298-47D5-AABB-2E7D00D2DECB.calendar
drwxr-xr-x  4 elwoodblues  staff     136 Sep 23 11:28 93B25A30-19AF-46BB-8416-AE184B886E00.calendar
drwxr-xr-x  3 elwoodblues  staff     102 Oct  4 13:03 Attachments
-rw-r--r--@ 1 elwoodblues  staff  217088 Oct  4 13:04 Calendar Cache
drwxr-xr-x  2 elwoodblues  staff      68 Oct  4 13:04 Calendar Sync Changes
```

```
Elwoods-Mac:Calendars elwoodblues$ pwd
/Users/elwoodblues/Library/Calendars
Elwoods-Mac:Calendars elwoodblues$ ls -laR 788EEBAA-B298-47D5-AABB-2E7D00D2DECB.calendar/
total 8
drwxr-xr-x  4 elwoodblues  staff  136 Sep 23 11:28 .
drwxr-xr-x  7 elwoodblues  staff  238 Oct  4 13:04 ..
drwxr-xr-x  3 elwoodblues  staff  102 Oct  4 13:03 Events
-rw-r--r--  1 elwoodblues  staff  565 Oct  4 13:03 Info.plist

788EEBAA-B298-47D5-AABB-2E7D00D2DECB.calendar//Events:
total 8
drwxr-xr-x  3 elwoodblues  staff  102 Oct  4 13:03 .
drwxr-xr-x  4 elwoodblues  staff  136 Sep 23 11:28 ..
-rw-r--r--  1 elwoodblues  staff  421 Oct  4 13:03 C5E7B415-079F-4BDB-81B9-1ECC8B483590.ics
```


macOS: `~/Library/Calendars/`
iOS Physical: `/private/var/mobile/Library/Calendar/`
iOS Backup: `/mobile/Library/Calendar/`

Each calendar is saved as a separate directory, which is named after the calendar GUID and ends with a `.calendar` or `.caldav` extension. Each calendar directory contains an `Events` directory and an `Info.plist`.

The `Info.plist` contains information about the calendar, such as calendar name, GUI preferences, and other configurable items.

The `Events` directory contains the calendar `.ics` files. These files contain the calendar entries. Each `.ics` file contains information for a single calendar event.

© 2020 Sarah Edwards

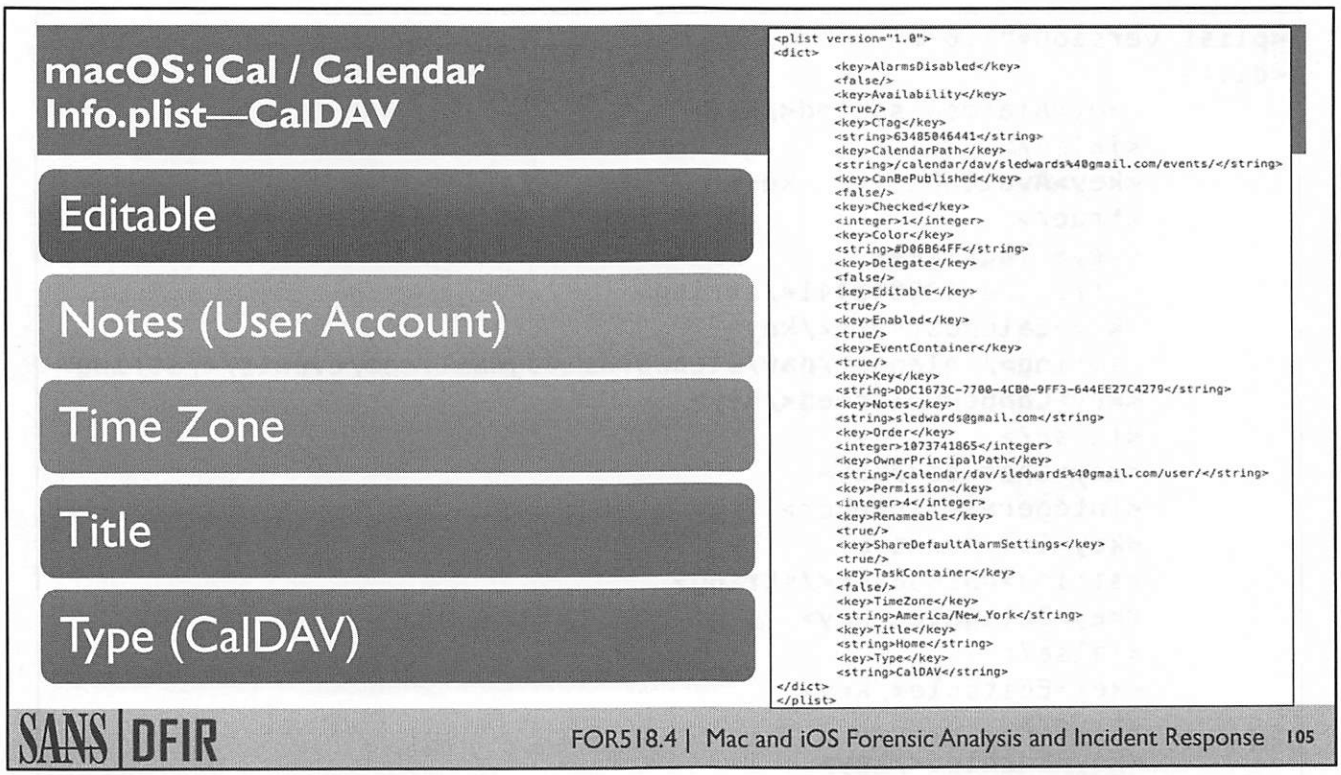
macOS: iCal / Calendar—Info.plist

```
Elwoods-Mac:788EEBAA-B298-47D5-AABB-2E7D00D2DECB.calendar elwoodblues$ cat Info.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd
">
<plist version="1.0">
<dict>
        <key>AlarmsDisabled</key>
        <false/>
        <key>Checked</key>
        <integer>1</integer>
        <key>Color</key>
        <string>#44A703FF</string>
        <key>Editable</key>
        <true/>
        <key>Enabled</key>
        <true/>
        <key>Key</key>
        <string>788EEBAA-B298-47D5-AABB-2E7D00D2DECB</string>
        <key>Order</key>
        <integer>2</integer>
        <key>Title</key>
        <string>Work</string>
        <key>Type</key>
        <string>Local</string>
</dict>
</plist>
```


In the example `Info.plist` above, the calendar is named "`Work`". This calendar has a color associated with it (a lovely green color if you look up HTML color codes).

Other configurable items include alarm preferences, if the calendar can be edited, and the type of calendar. This one happens to be a local calendar, rather than a CalDAV calendar.

© 2020 Sarah Edwards



This slide shows an example of a CalDAV calendar `Info.plist`. CalDAV is a standard that allows multiple applications, such as iCal or Google Calendar, to save the same information and keep it synced between clients.

The CalDAV `Info.plist` file may contain more information than a normal calendar `Info.plist` file.

- Account Information, such as Google Calendars or iCloud accounts
- Time Zone Information
- Last Sync Date
- Sync Settings

© 2020 Sarah Edwards

```
<plist version="1.0">
<dict>
        <key>AlarmsDisabled</key>
        <false/>
        <key>Availability</key>
        <true/>
        <key>CTag</key>
        <string>63485046441</string>
        <key>CalendarPath</key>
        <string>/calendar/dav/sledwards%40gmail.com/events/</string>
        <key>CanBePublished</key>
        <false/>
        <key>Checked</key>
        <integer>1</integer>
        <key>Color</key>
        <string>#D06B64FF</string>
        <key>Delegate</key>
        <false/>
        <key>Editable</key>
        <true/>
        <key>Enabled</key>
        <true/>
        <key>EventContainer</key>
        <true/>
        <key>Key</key>
        <string>DDC1673C-7700-4CB0-9FF3-644EE27C4279</string>
        <key>Notes</key>
        <string>sledwards@gmail.com</string>
        <key>Order</key>
        <integer>1073741865</integer>
        <key>OwnerPrincipalPath</key>
        <string>/calendar/dav/sledwards%40gmail.com/user/</string>
        <key>Permission</key>
        <integer>4</integer>
        <key>Renameable</key>
        <true/>
        <key>ShareDefaultAlarmSettings</key>
        <true/>
        <key>TaskContainer</key>
        <false/>
        <key>TimeZone</key>
        <string>America/New_York</string>
        <key>Title</key>
        <string>Home</string>
        <key>Type</key>
        <string>CalDAV</string>
</dict>
</plist>
```


© 2020 Sarah Edwards

iCal / Calendar: Events – *.ics Files

```
Elwoods-Mac:Events elwoodblues$ cat C5E7B415-079F-4BDB-81B9-1ECC8B483590.ics
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Apple Inc.//iCal 5.0//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
CREATED:20121004T180254Z
UID:C5E7B415-079F-4BDB-81B9-1ECC8B483590
DTEND;TZID=America/Chicago:20121004T160000
TRANSP:OPAQUE
X-APPLE-DONTSCHEDULE:TRUE
SUMMARY:Meeting with Jake
DTSTART;TZID=America/Chicago:20121004T150000
DTSTAMP:20121004T180325Z
X-APPLE-EWS-BUSYSTATUS:BUSY
SEQUENCE:2
END:VEVENT
END:VCALENDAR
```


Each `.ics` file contained within a calendar is a calendar event. Shown above, it may contain:

- Create Date
- Event Start Time
- Event End Time
- Unique ID
- Time Zone Information
- Event Summary

© 2020 Sarah Edwards

```
Elwoods-Mac:Events elwoodblues$ cat C5E7B415-079F-4BDB-81B9-1ECC8B483590.ics
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Apple Inc.//iCal 5.0//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
CREATED:20121004T180254Z
UID:C5E7B415-079F-4BDB-81B9-1ECC8B483590
DTEND;TZID=America/Chicago:20121004T160000
TRANSP:OPAQUE
X-APPLE-DONTSCHEDULE:TRUE
SUMMARY:Meeting with Jake
DTSTART;TZID=America/Chicago:20121004T150000
DTSTAMP:20121004T180325Z
X-APPLE-EWS-BUSYSTATUS:BUSY
SEQUENCE:2
END:VEVENT
END:VCALENDAR
```

© 2020 Sarah Edwards

iCal / Calendar: CalDAV Events – *.ics Files

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Apple Inc.//Mac OS X 10.8//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
DTEND;VALUE=DATE:20101202
TRANSP:TRANSPARENT
UID:vaoav53oi9f24vtam3lgf2s7po@google.com
DTSTAMP:20101102T005302Z
LOCATION:
DESCRIPTION:
STATUS:CONFIRMED
X-APPLE-SCHEDULETAG:
X-APPLE-SERVERFILENAME:vaoav53oi9f24vtam3lgf2s7po%40google.com.ics
SEQUENCE:0
X-APPLE-EWS-BUSYSTATUS:FREE
SUMMARY:Shmoocon Tickets
LAST-MODIFIED:20101102T005302Z
DTSTART;VALUE=DATE:20101201
CREATED:20101020T200408Z
X-APPLE-ETAG:"63424342382"
BEGIN:VALARM
X-WR-ALARMUID:C0E39F79-2073-4050-B553-4674EF592ADA
UID:C0E39F79-2073-4050-B553-4674EF592ADA
TRIGGER;VALUE=DATE-TIME:20101130T215000Z
DESCRIPTION:This is an event reminder
ACTION:DISPLAY
END:VALARM
BEGIN:VALARM
X-WR-ALARMUID:BE34309D-161E-420B-8418-537F3779CB14
UID:BE34309D-161E-420B-8418-537F3779CB14
TRIGGER:-PT15H
X-APPLE-DEFAULT-ALARM:TRUE
ATTACH;VALUE=URI:Basso
ACTION:AUDIO
END:VALARM
END:VEVENT
END:VCALENDAR
```


CalDAV calendar events may have additional information, depending on the client the event was created with. The example above shows a Google Calendar event.

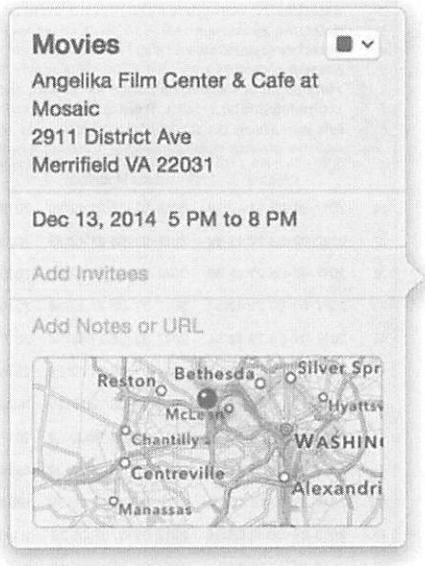
© 2020 Sarah Edwards

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Apple Inc.//Mac OS X 10.8//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
DTEND;VALUE=DATE:20101202
TRANSP:TRANSPARENT
UID:vaoav53oi9f24vtam3lgf2s7po@google.com
DTSTAMP:20101102T005302Z
LOCATION:
DESCRIPTION:
STATUS:CONFIRMED
X-APPLE-SCHEDULETAG:
X-APPLE-SERVERFILENAME:vaoav53oi9f24vtam3lgf2s7po%40google.com.ics
SEQUENCE:0
X-APPLE-EWS-BUSYSTATUS:FREE
SUMMARY:Shmoocon Tickets
LAST-MODIFIED:20101102T005302Z
DTSTART;VALUE=DATE:20101201
CREATED:20101020T200408Z
X-APPLE-ETAG:"63424342382"
BEGIN:VALARM
X-WR-ALARMUID:C0E39F79-2073-4050-B553-4674EF592ADA
UID:C0E39F79-2073-4050-B553-4674EF592ADA
TRIGGER;VALUE=DATE-TIME:20101130T215000Z
DESCRIPTION:This is an event reminder
ACTION:DISPLAY
END:VALARM
BEGIN:VALARM
X-WR-ALARMUID:BE34309D-161E-420B-8418-537F3779CB14
UID:BE34309D-161E-420B-8418-537F3779CB14
TRIGGER:-PT15H
X-APPLE-DEFAULT-ALARM:TRUE
ATTACH;VALUE=URI:Basso
ACTION:AUDIO
END:VALARM
END:VEVENT
END:VCALENDAR
```


© 2020 Sarah Edwards

iCal / Calendar: Events with Location

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Apple Inc.//Mac OS X 10.10.1//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
TRANSP:OPAQUE
DTEND;TZID=America/New_York:20141213T200000
X-APPLE-STRUCTURED-LOCATION;VALUE=URI;X-ADDRESS=2911 District Ave\\nMerr
 ifield VA 22031;X-APPLE-RADIUS=14161.30926963496;X-TITLE=Angelika Film C
 enter & Cafe at Mosaic:geo:38.872212,-77.229711
UID:61BE4907-3EA5-456E-95C6-6552322AF6C2
DTSTAMP:20141213T172708Z
LOCATION:Angelika Film Center & Cafe at Mosaic\n2911 District Ave\nMerri
 field VA 22031
DESCRIPTION:
STATUS:CONFIRMED
X-APPLE-SCHEDULETAG:"87756a35d7205eb3"
X-APPLE-SERVERFILENAME:61BE4907-3EA5-456E-95C6-6552322AF6C2.ics
SEQUENCE:0
X-APPLE-EWS-BUSYSTATUS:BUSY
SUMMARY:Movies
DTSTART;TZID=America/New_York:20141213T170000
LAST-MODIFIED:20141213T172708Z
CREATED:20141213T172708Z
X-APPLE-ETAG:"63554174828"
END:VEVENT
END:VCALENDAR
```



Calendar events may also be created with location information.

In the screenshot above, a calendar entry was created at a particular theater in Merrifield, VA. The user is able to search and select a particular location, which is embedded in the ICS file, as shown above.

© 2020 Sarah Edwards

macOS Calendar Cache (iOS Calendar.sqlitedb): Calendar Events

```
1  select
2  datetime(zcreationdate+978307200,'unixepoch','localtime') as "Created",
3  datetime(zdatestamp+978307200,'unixepoch','localtime') as "Last Modified",
4  datetime(zstartdate+978307200,'unixepoch','localtime') as "Event Start",
5  datetime(zenddate+978307200,'unixepoch','localtime') as "Event End",
6  ztimezone, zcalendaritem.ztitle as "Event", znode.ztitle as "Calendar/List" ,
7  zcompleteddate, zstatus, from zcalendaritem
8  left join znode on zcalendaritem.zcalendar = znode.z_pk
```

	Created	Last Modified	Event Start	Event End	ZTIMEZONE	Event	Calendar/Reminder List
36	2017-02-08 21:42:39	2017-02-08 21:42:39	2016-03-17 19:00:00	2016-03-17 20:30:00	America/New_...	Reservation: Charley's Ste...	Found in Apps
37	2017-02-08 21:42:39	2017-02-08 21:42:39	2016-03-13 16:00:00	2016-03-13 17:30:00	GMT-0700	Reservation: Skillet Diner - ...	Found in Apps
38	2017-02-08 21:42:36	2017-02-08 21:42:36	2016-05-31 19:00:00	2016-05-31 20:30:00	GMT-0400	Reservation: Texas Jacks ...	Found in Apps
39	2017-02-08 21:42:34	2017-02-08 21:42:34	2016-08-03 21:00:00	2016-08-04 00:00:00	GMT-0700	Ticket: Volatility Black Hat ...	Found in Apps
40	2017-02-08 21:42:34	2017-02-08 21:42:34	2016-07-04 16:00:00	2016-07-04 17:30:00	America/New_...	Ticket: 4th of July BBQ	Found in Apps
41	2017-02-08 21:42:33	2017-02-08 21:42:33	2016-08-13 12:00:00	2016-08-13 13:30:00	GMT-0400	Ticket: MacDMV August M...	Found in Apps
42	2017-02-08 19:27:20	2017-02-08 19:27:20	1904-06-17 19:00:00	1904-06-18 19:00:00	NULL	[redacted]	Birthdays
43	2017-02-06 18:56:56	2017-02-08 16:35:22	2017-02-24 20:00:00	2017-02-24 22:00:00	GMT	[redacted]	Facebook Events
44	2017-02-08 15:38:32	2017-02-08 15:38:32	1984-09-06 20:00:00	1984-09-07 20:00:00	NULL	[redacted]	Birthdays
45	2017-02-07 19:45:29	2017-02-07 19:45:29	1904-09-26 19:00:00	1904-09-27 19:00:00	NULL	[redacted]	Birthdays
46	2013-08-08 12:00:55	2017-02-07 18:01:28	2016-01-12 19:00:00	2016-01-13 19:00:00	NULL	[redacted]	Home
47	2017-02-07 13:00:10	2017-02-07 13:00:10	2017-03-24 10:30:00	2017-03-24 11:30:00	US/Eastern	[redacted]	Home

SANS | DFIR FOR518.4 | Mac and iOS Forensic Analysis and Incident Response 112

The `Calendar Cache` (or `Calendar.sqlitedb` on iOS) SQLite database contains information for the Calendars. These databases have slightly different table names and have also changed column names over time, but contain generally the same data.

The `ZCALENDARITEM` (previously `ZICSELEMENT`) table contains calendar event information. Each tuple contains data similar to that found in the `.ics` file.

These calendar items or reminders contain a variety of data, partially shown here for brevity.

Items included are:

- Summary/item name
- Start and end dates
- If the calendar item is an all-day event
- What calendar it belongs to (review the Calendar table)
- When the event or reminder was last modified/created
- When the reminder was completed
- … so much more!

```
select
datetime(zcreationdate+978307200,'unixepoch','localtime') as "Created",
datetime(zdatestamp+978307200,'unixepoch','localtime') as "Last Modified",
datetime(zstartdate+978307200,'unixepoch','localtime') as "Event Start",
datetime(zenddate+978307200,'unixepoch','localtime') as "Event End",
ztimezone,
zcalendaritem.ztitle as "Event", znode.ztitle as "Calendar/List" ,
zcompleteddate, zstatus
from zcalendaritem
left join znode on zcalendaritem.zcalendar = znode.z_pk
```


© 2020 Sarah Edwards

```
1  select
2  datetime(zcreationdate+978307200,'unixepoch','localtime') as "Created",
3  datetime(zdatestamp+978307200,'unixepoch','localtime') as "Last Modified",
4  datetime(zstartdate+978307200,'unixepoch','localtime') as "Event Start",
5  datetime(zenddate+978307200,'unixepoch','localtime') as "Event End",
6  ztimezone, zcalendaritem.ztitle as "Event", znode.ztitle as "Calendar/List" ,
7  zcompleteddate, zstatus, from zcalendaritem
8  left join znode on zcalendaritem.zcalendar = znode.z_pk
```

	Created	Last Modified	Event Start	Event End	ZTIMEZONE	Event	Calendar/Reminder List
36	2017-02-08 21:42:39	2017-02-08 21:42:39	2016-03-17 19:00:00	2016-03-17 20:30:00	America/New_...	Reservation: Charley's Ste...	Found in Apps
37	2017-02-08 21:42:39	2017-02-08 21:42:39	2016-03-13 16:00:00	2016-03-13 17:30:00	GMT-0700	Reservation: Skillet Diner - ...	Found in Apps
38	2017-02-08 21:42:36	2017-02-08 21:42:36	2016-05-31 19:00:00	2016-05-31 20:30:00	GMT-0400	Reservation: Texas Jacks ...	Found in Apps
39	2017-02-08 21:42:34	2017-02-08 21:42:34	2016-08-03 21:00:00	2016-08-04 00:00:00	GMT-0700	Ticket: Volatility Black Hat ...	Found in Apps
40	2017-02-08 21:42:34	2017-02-08 21:42:34	2016-07-04 16:00:00	2016-07-04 17:30:00	America/New_...	Ticket: 4th of July BBQ	Found in Apps
41	2017-02-08 21:42:33	2017-02-08 21:42:33	2016-08-13 12:00:00	2016-08-13 13:30:00	GMT-0400	Ticket: MacDMV August M...	Found in Apps
42	2017-02-08 19:27:20	2017-02-08 19:27:20	1904-06-17 19:00:00	1904-06-18 19:00:00	*NULL*	[redacted]	Birthdays
43	2017-02-06 18:56:56	2017-02-08 16:35:22	2017-02-24 20:00:00	2017-02-24 22:00:00	GMT	[redacted]	Facebook Events
44	2017-02-08 15:38:32	2017-02-08 15:38:32	1984-09-06 20:00:00	1984-09-07 20:00:00	*NULL*	[redacted]	Birthdays
45	2017-02-07 19:45:29	2017-02-07 19:45:29	1904-09-26 19:00:00	1904-09-27 19:00:00	*NULL*	[redacted]	Birthdays
46	2013-08-08 12:00:55	2017-02-07 18:01:28	2016-01-12 19:00:00	2016-01-13 19:00:00	*NULL*	[redacted]	Home
47	2017-02-07 13:00:10	2017-02-07 13:00:10	2017-03-24 10:30:00	2017-03-24 11:30:00	US/Eastern	[redacted]	Home

© 2020 Sarah Edwards

macOS Calendar Cache (iOS Calendar.sqlitedb): Reminder Items


```
1  select
2  datetime(zcreationdate+978307200,'unixepoch','localtime') as "Created",
3  datetime(zdatestamp+978307200,'unixepoch','localtime') as "Last Modified",
4  datetime(zstartdate+978307200,'unixepoch','localtime') as "Event Start",
5  datetime(zenddate+978307200,'unixepoch','localtime') as "Event End",
6  ztimezone, zcalendaritem.ztitle as "Event", znode.ztitle as "Calendar/List" ,
7  zcompleteddate, zstatus, from zcalendaritem
8  left join znode on zcalendaritem.zcalendar = znode.z_pk
```

	Created	Last Modified	Event Start	Event End	ZTIMEZONE	Event	Calendar/Reminder List	ZCOMPLETEDDATE	ZSTATUS
1	2017-02-11 20:10:43	2017-02-11 21:27:44	NULL	NULL	NULL	Guac	Shopping	508559264	COMPLETED
2	2017-02-11 20:06:27	2017-02-11 21:27:41	NULL	NULL	NULL	Cookies	Shopping	508559261	COMPLETED
3	2017-02-11 20:10:47	2017-02-11 21:08:10	NULL	NULL	NULL	Coke	Shopping	NULL	NULL
4	2017-02-11 20:10:45	2017-02-11 20:10:46	NULL	NULL	NULL	Chips	Shopping	NULL	NULL
5	2017-02-11 20:06:25	2017-02-11 20:06:26	NULL	NULL	NULL	Red Bull	Shopping	NULL	NULL

macOS: `~/Library/Calendars/`
iOS Physical: `/private/var/mobile/Library/Calendar/`
iOS Backup: `/mobile/Library/Calendar/`

The Reminders application also uses the (`Calendar Cache/Calendar.sqlitedb`) file.

Each reminder may be associated with a specific list; in this example, Shopping. This list's names can be correlated with data found in the `ZNODE` table of this database.

A creation timestamp is recorded when the list item is put into the application. After the item is specified as completed by the user, a completed date is recorded and the status set to "`COMPLETED`". While these items no longer show up in the current list, an investigator may still see them in the database.

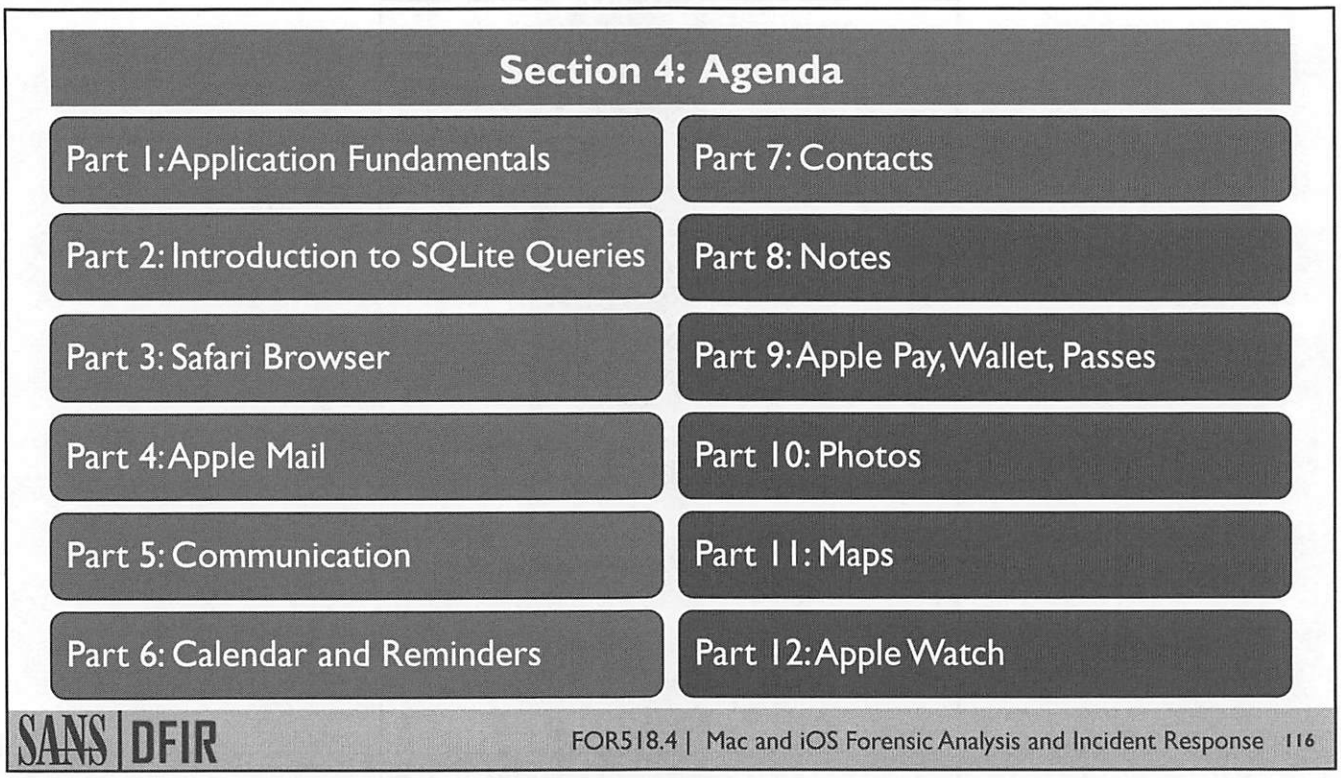
```
select
datetime(zcreationdate+978307200,'unixepoch','localtime') as "Created",
datetime(zdatestamp+978307200,'unixepoch','localtime') as "Last Modified",
datetime(zstartdate+978307200,'unixepoch','localtime') as "Event Start",
datetime(zenddate+978307200,'unixepoch','localtime') as "Event End",
ztimezone,
zcalendaritem.ztitle as "Event", znode.ztitle as "Calendar/List",
zcompleteddate, zstatus from zcalendaritem
left join znode on zcalendaritem.zcalendar = znode.z_pk
```


© 2020 Sarah Edwards

```
1  select
2  datetime(zcreationdate+978307200,'unixepoch','localtime') as "Created",
3  datetime(zdatestamp+978307200,'unixepoch','localtime') as "Last Modified",
4  datetime(zstartdate+978307200,'unixepoch','localtime') as "Event Start",
5  datetime(zenddate+978307200,'unixepoch','localtime') as "Event End",
6  ztimezone, zcalendaritem.ztitle as "Event", znode.ztitle as "Calendar/List" ,
7  zcompleteddate, zstatus, from zcalendaritem
8  left join znode on zcalendaritem.zcalendar = znode.z_pk
```

	Created	Last Modified	Event Start	Event End	ZTIMEZONE	Event	Calendar/Reminder List	ZCOMPLETEDDATE	ZSTATUS
1	2017-02-11 20:10:43	2017-02-11 21:27:44	*NULL*	*NULL*	*NULL*	Guac	Shopping	508559264	COMPLETED
2	2017-02-11 20:06:27	2017-02-11 21:27:41	*NULL*	*NULL*	*NULL*	Cookies	Shopping	508559261	COMPLETED
3	2017-02-11 20:10:47	2017-02-11 21:08:10	*NULL*	*NULL*	*NULL*	Coke	Shopping	*NULL*	*NULL*
4	2017-02-11 20:10:45	2017-02-11 20:10:46	*NULL*	*NULL*	*NULL*	Chips	Shopping	*NULL*	*NULL*
5	2017-02-11 20:06:25	2017-02-11 20:06:26	*NULL*	*NULL*	*NULL*	Red Bull	Shopping	*NULL*	*NULL*

© 2020 Sarah Edwards



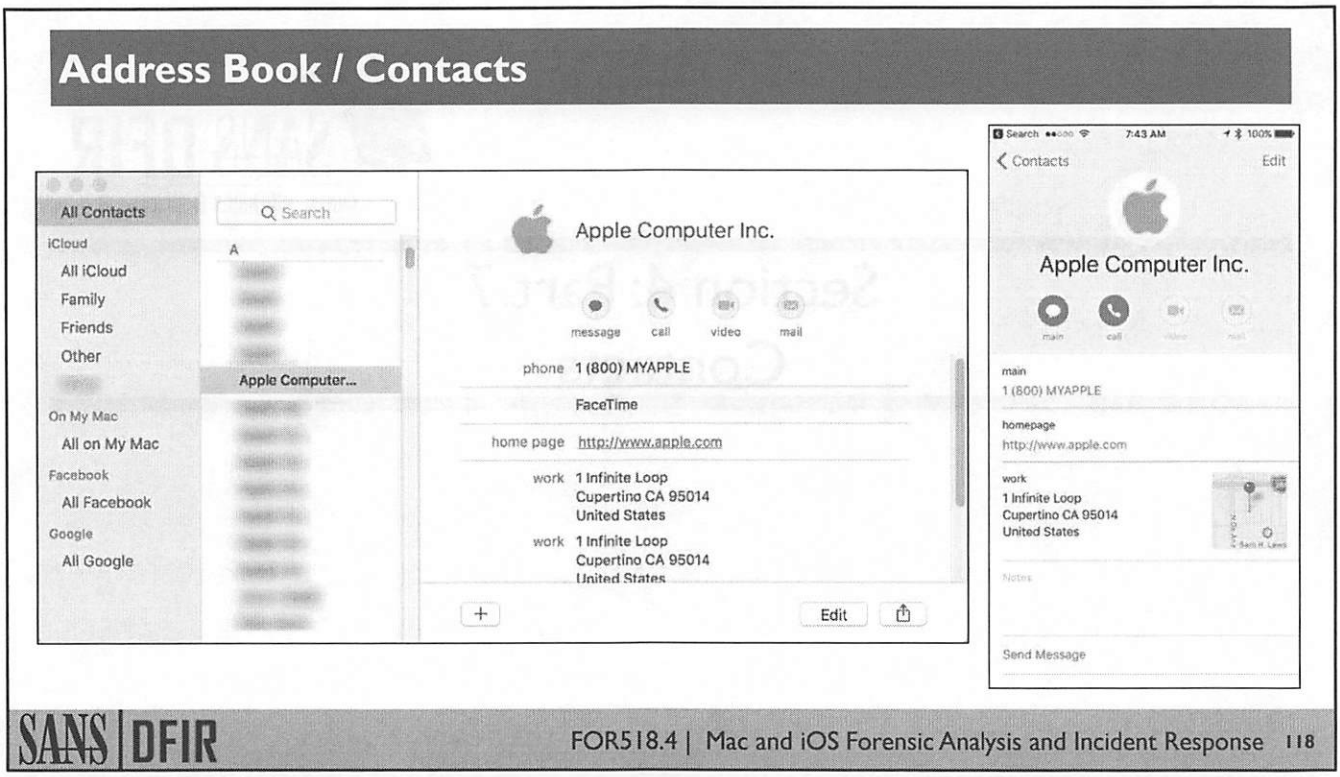
This page intentionally left blank.

© 2020 Sarah Edwards

Section 4: Part 7
Contacts

This page intentionally left blank.

© 2020 Sarah Edwards



macOS: `~/Library/Application Support/AddressBook/`
iOS Physical: `/private/var//mobile/Library/AddressBook/`
iOS Backup: `/mobile/Library/AddressBook/`

The Address Book, or Contacts application, holds the user's contact information. This database gets populated by the user but also "automagically" from other sources if the user associates their email and social media accounts with the operating system.

The left-hand side of the macOS screenshot shows contacts from iCloud, local (On My Mac), Facebook, and Google.

Just like iCal/Calendar, the Address Book was renamed to Contacts in Mountain Lion (10.8).

© 2020 Sarah Edwards

macOS Address Book / Contacts: Different Sources ~/Library/Application Support/AddressBook/Sources/

```
bit:Sources oompa$ pwd
/Users/oompa/Library/Application Support/AddressBook/Sources
bit:Sources oompa$ ls -l
total 0
drwxr-xr-x  10 oompa  staff  340 Feb  6 18:57 BF7B48B9-0A83-4190-9F64-BB8779BA6128
drwxr-xr-x  11 oompa  staff  374 Feb 12 07:39 ED106899-0EA3-4DAB-86AB-83DDF17B159C
drwxr-xr-x  10 oompa  staff  340 Feb  6 18:57 EE54CA64-23CD-4CFB-9D8D-EB96FC79E8E2
bit:Sources oompa$ ls -l ED106899-0EA3-4DAB-86AB-83DDF17B159C/
total 5920
-rw-r--r--@   1 oompa  staff   962560 Feb  6 18:57 AddressBook-v22.abcddb
-rw-r--r--@   1 oompa  staff    32768 Feb  7 18:05 AddressBook-v22.abcddb-shm
-rw-r--r--@   1 oompa  staff  2031192 Feb 12 07:39 AddressBook-v22.abcddb-wal
drwx------   43 oompa  staff     1462 Feb  6 07:48 Images
drwx------  376 oompa  staff    12784 Feb 12 07:39 Metadata
-rw-r--r--@   1 oompa  staff      181 Feb 12 07:39 OfflineDeletedItems.plist
-rwxr-xr-x@   1 oompa  staff        0 Feb  6 07:48 OfflineDeletedItems.plist.lockfile
-rwxr-xr-x@   1 oompa  staff        0 Feb  6 07:48 Sync.lockfile
-rwxr-xr-x@   1 oompa  staff        0 Feb  6 07:48 SyncOperations.plist.lockfile
```


macOS: `~/Library/Application Support/AddressBook/`
iOS Physical: `/private/var//mobile/Library/AddressBook/`
iOS Backup: `/mobile/Library/AddressBook/`

Each Address Book database under Sources will have contact information separated out. One could be the contacts for Facebook, Google, iCloud—or another source.

These sources can be determined by matching the GUID up in the `Accounts` database.

© 2020 Sarah Edwards

macOS Address Book / Contacts: Metadata Directories

*.abcdp: Per Person

*.abcdg: Per Group

*.abcds: One Subscription Record (hidden file)

```
[bit:Metadata oompa$ pwd
/Users/oompa/Library/Application Support/AddressBook/Sources/ED106899-0EA3-4DAB-86AB-83DDF17B159C/Metadata
[bit:Metadata oompa$ ls -l
total 3016
-rw-------  1 oompa  staff   1374 Feb  6 07:48 004F7096-2B53-43B0-9854-B543ABB8AA8E:ABPerson.abcdp
-rw-------@ 1 oompa  staff   1895 Feb  7 08:39 006A5B28-152C-4EAB-805C-40EB05B5C861:ABPerson.abcdp
-rw-------  1 oompa  staff   1366 Feb  6 18:58 0128C2A0-07F1-4E41-949A-B62C7900260B:ABPerson.abcdp
-rw-------  1 oompa  staff   1391 Feb  6 07:48 019FB7C2-1FA1-4001-A015-D53C7456C3E6:ABPerson.abcdp
```


The Metadata directory contains a file for each person, group, or subscription. Each file is named accordingly: "p" for person, "g" for group, and "s" for subscription. Each file is a binary property list containing the information for that particular Address Book entry.

Much of the same information found in the Metadata directory may be found in the SQLite database `AddressBook-v22.abcddb`, in the `~/Library/Application Support/AddressBook/` directory.

Newer versions of OS X have a Sources directory that contains the GUID of the specific Address Book source. You will find the Metadata directory here.

© 2020 Sarah Edwards

macOS Address Book / Contacts: Person Record Metadata Directory – *.abcdp

- Timestamps (Creation, Modification)
- Contact Name
- Contact Address Book GUID
- Email Addresses
- Physical Address
- Phone Numbers
- Social Media Accounts
- Websites
- Birthday
- Etc.

Key	Type	Value
▼Root	Dictionary	(10 items)
UID	String	5F7FB365-A760-4486-B60E-03318C40376A:ABPerson
Creation	Date	Sep 23, 2012 12:29:43 PM
First	String	Elwood
Modification	Date	Oct 5, 2012 9:12:54 PM
▼Phone	Dictionary	(4 items)
▶identifiers	Array	(1 item)
▼values	Array	(1 item)
Item 0	String	515 5551212
primary	String	408E586B-3CCE-44E7-AC4D-5C69D6C33E1D
▶labels	Array	(1 item)
ABPersonFlags	Number	0
▼Address	Dictionary	(4 items)
▶identifiers	Array	(1 item)
▼values	Array	(1 item)
▼Item 0	Dictionary	(6 items)
Street	String	1060 W Addison
ZIP	String	60613
CountryCode	String	us
City	String	Chicago
State	String	IL
Country	String	United States
primary	String	6DB3E8ED-5618-4204-AA8E-58F9A2AD8AF4
▶labels	Array	(1 item)
▶ABPropertyTypes	Dictionary	(37 items)
Last	String	Blues
▶Email	Dictionary	(4 items)

To view the raw Address Book records with Xcode, you will need to rename them as `.plist`; otherwise, they will show up in a view similar to what is found in the Address Book/Contacts application.

Each `Person` record has its own GUID in the `UID` key. This GUID may be used in the `Group` record if the `Person` record is part of a group.

The `Creation` key contains the date the record was created, while the `Modification` date contains the date when the contact was last modified.

Each `Person` record contains the phone, address, email, or other information the user has supplied. If the contact has multiple entries under the "`values`" key, such as multiple phone numbers, the "`primary`" key will determine which of these is listed as the primary contact number.

© 2020 Sarah Edwards

Key	Type	Value
▼ Root	Dictionary	(10 items)
UID	String	5F7FB365-A760-4486-B60E-03318C40376A:ABPerson
Creation	Date	Sep 23, 2012 12:29:43 PM
First	String	Elwood
Modification	Date	Oct 5, 2012 9:12:54 PM
▼ Phone	Dictionary	(4 items)
▶ identifiers	Array	(1 item)
▼ values	Array	(1 item)
Item 0	String	515 5551212
primary	String	408E5B6B-3CCE-44E7-AC4D-5C69D6C33E1D
▶ labels	Array	(1 item)
ABPersonFlags	Number	0
▼ Address	Dictionary	(4 items)
▶ identifiers	Array	(1 item)
▼ values	Array	(1 item)
▼ Item 0	Dictionary	(6 items)
Street	String	1060 W Addison
ZIP	String	60613
CountryCode	String	us
City	String	Chicago
State	String	IL
Country	String	United States
primary	String	6DB3EBED-561B-4204-AA8E-58F9A2AD8AF4
▶ labels	Array	(1 item)
▶ ABPropertyTypes	Dictionary	(37 items)
Last	String	Blues
▶ Email	Dictionary	(4 items)

© 2020 Sarah Edwards

macOS Address Book / Contacts: Group Record Metadata Directory – *.abcdg

Key	Type	Value
▼ Root	Dictionary	(8 items)
UID	String	C45523F6-04BD-44FB-8574-0042F21D4633:ABGroup
ABGroupClassKey	String	ABGroup
▶ ABAddressDistributionList	Dictionary	(0 items)
ABPersonFlags	Number	0
▶ ABEmailDistributionList	Dictionary	(0 items)
▼ ABMembers	Array	(2 items)
Item 0	String	5F7FB365-A760-4486-B60E-03318C40376A:ABPerson
Item 1	String	9BBD4905-88A0-4201-9E75-C61E6FB92E77:ABPerson
▶ ABPhoneDistributionList	Dictionary	(0 items)
GroupName	String	Family

The `Group` record contains all the GUIDs of `Person` records that are part of that group in the `ABMembers` key. Each group also has its own GUID that can be referenced by various applications.

The keys `ABEmailDistributionList`, `ABPhoneDistributionList`, and `ABAddressDistributionList` may be populated if the records are part of an email, phone, or mailing distribution list.

© 2020 Sarah Edwards

Key	Type	Value
▼ Root	Dictionary	(8 items)
UID	String	C45523F6-04BD-44FB-8574-0042F21D4633:ABGroup
ABGroupClassKey	String	ABGroup
▶ ABAddressDistributionList	Dictionary	(0 items)
ABPersonFlags	Number	0
▶ ABEmailDistributionList	Dictionary	(0 items)
▼ ABMembers	Array	(2 items)
Item 0	String	5F7FB365-A760-4486-B60E-03318C40376A:ABPerson
Item 1	String	9BBD4905-88A0-4201-9E75-C61E6FB92E77:ABPerson
▶ ABPhoneDistributionList	Dictionary	(0 items)
GroupName	String	Family

© 2020 Sarah Edwards

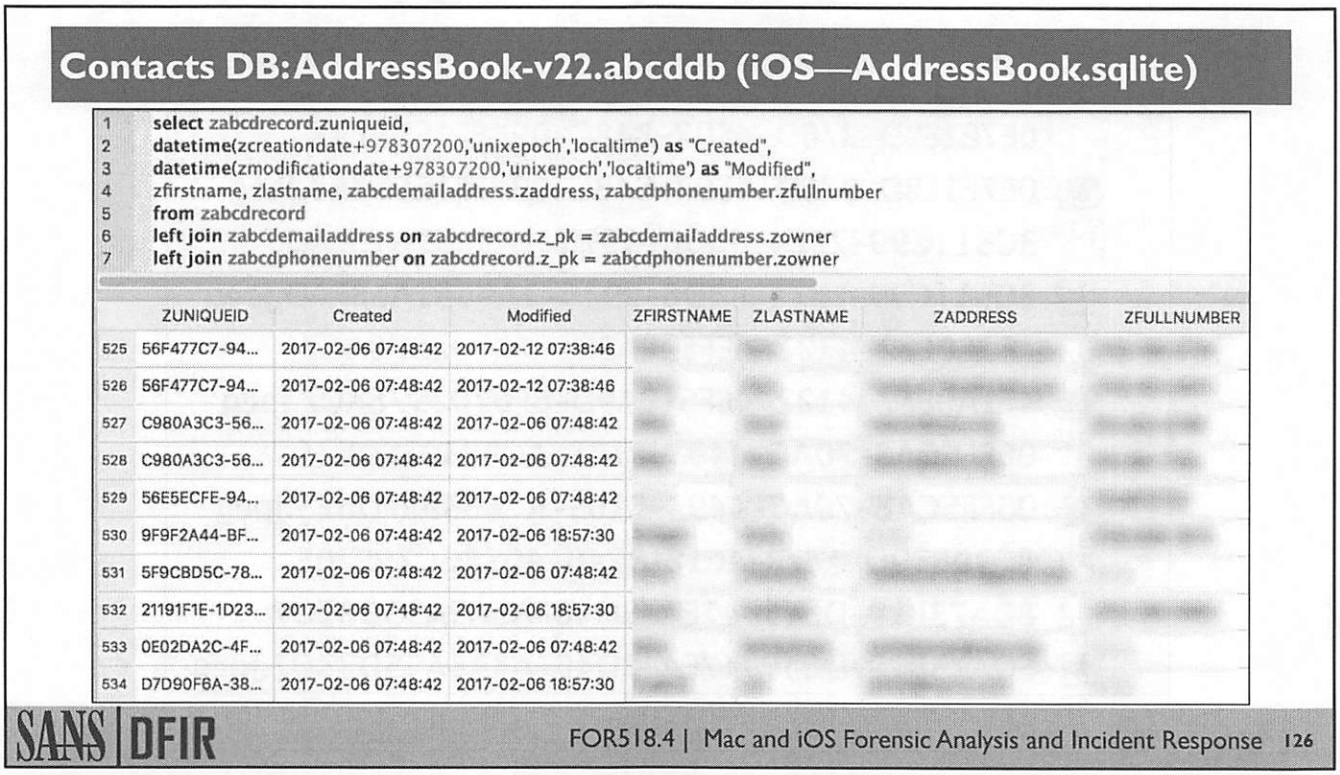
macOS Address Book / Contacts: Images Directory

The `Images` directory contains contact profile pictures named by that contact's GUID.

Some contacts may have more than one icon picture associated with them. This may be due to the pictures being resized or cropped to show well in the Address Book or Contacts application.

Newer versions of OS X have a `Sources` directory that contains the GUID of the specific Address Book source. You will find the `Images` directory here.

© 2020 Sarah Edwards



macOS: `~/Library/Application Support/AddressBook/AddressBook-v22.abcddb`
iOS Physical: `/private/var//mobile/Library/AddressBook/AddressBook.sqlite`
iOS Backup: `/mobile/Library/AddressBook/AddressBook.sqlite`

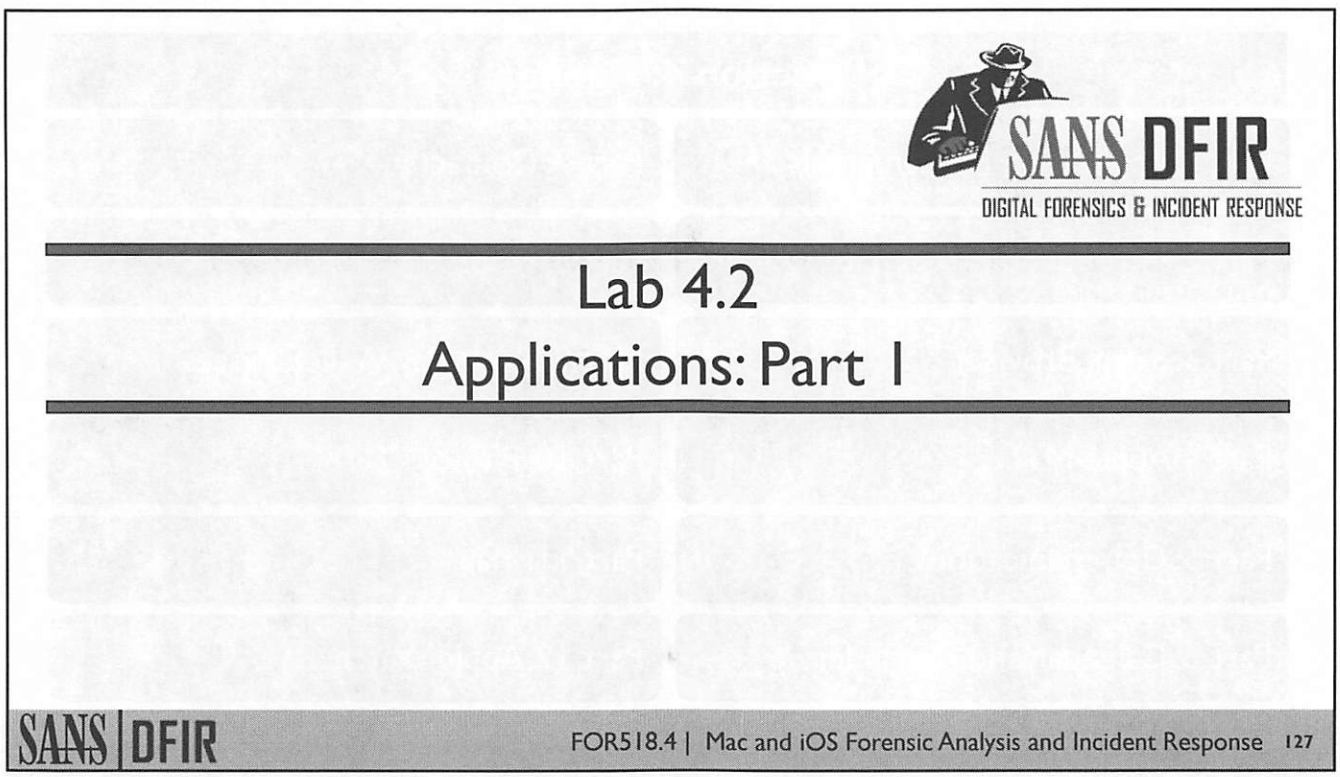
The Contacts application stores its data in SQLite databases, `AddressBook.sqlite` on iOS and `AddressBook-v22.abcddb` on macOS. The example shown above is from macOS 10.12. The table and column names from iOS and macOS will be slightly different but will contain the same type of information—a similar SQL query can be constructed to gather the same information or additional information as needed for the analyst.

The databases contain contacts information including names, emails, addresses, phone numbers, social media accounts, etc.

The `ABCDRECORD` table (`ABPerson` on iOS) stores the contact name, organization, job title, etc. This table also stores the contact creation and modification dates. There are many more tables that hold additional information. In the example, the email addresses and phone numbers were extracted from other tables.

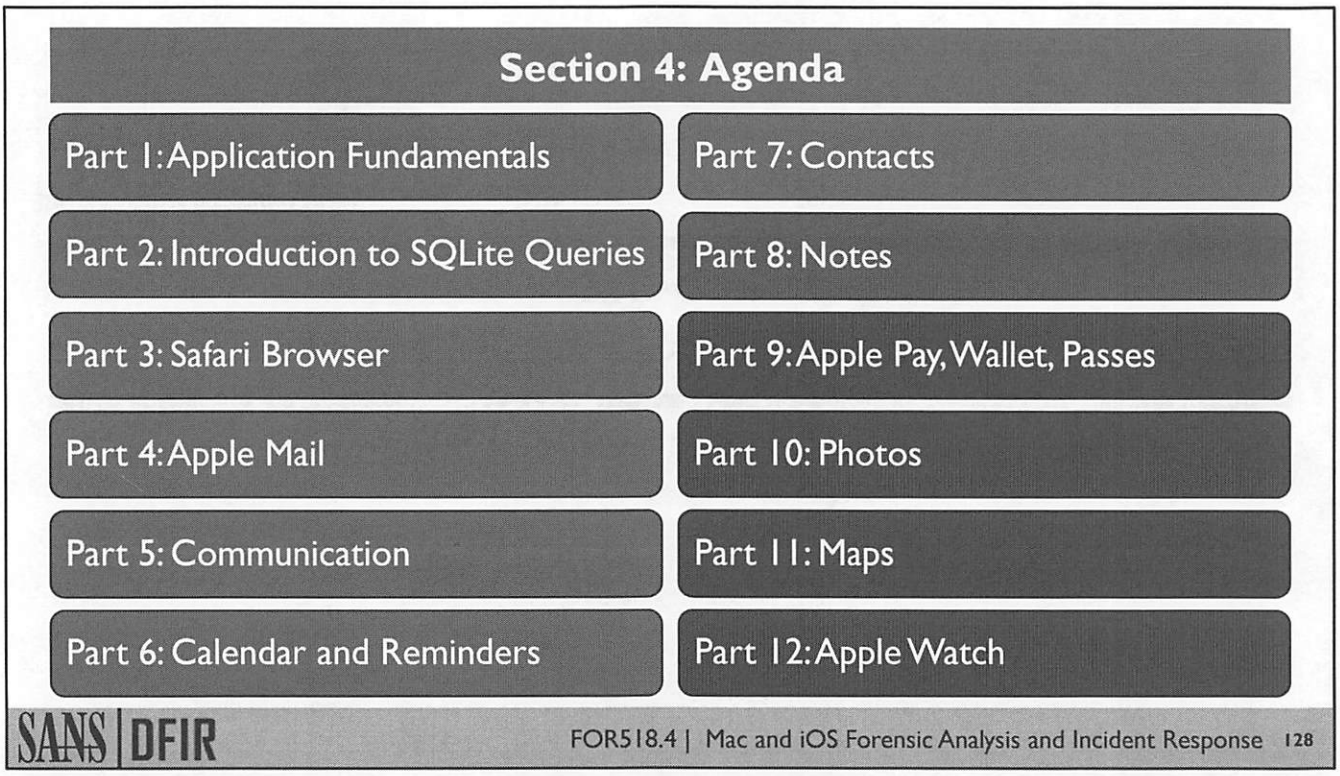
```
select
zabcdrecord.zuniqueid,
datetime(zcreationdate+978307200,'unixepoch','localtime') as "Created",
datetime(zmodificationdate+978307200,'unixepoch','localtime') as "Modified",
zfirstname,
zlastname,
zabcdemailaddress.zaddress,
zabcdphonenumber.zfullnumber
from zabcdrecord
left join zabcdemailaddress on zabcdrecord.z_pk = zabcdemailaddress.zowner
left join zabcdphonenumber on zabcdrecord.z_pk = zabcdphonenumber.zowner
```


© 2020 Sarah Edwards



This page intentionally left blank.

© 2020 Sarah Edwards



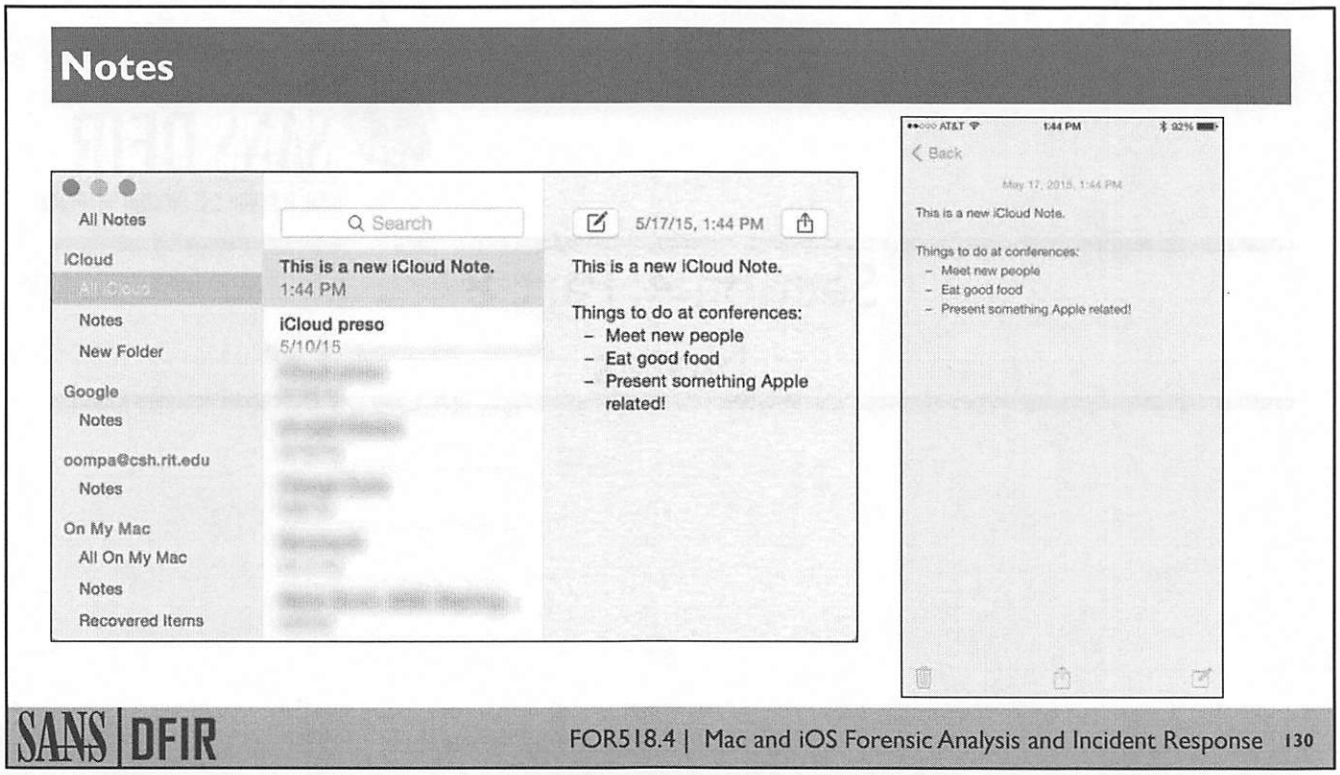
This page intentionally left blank.

© 2020 Sarah Edwards

Section 4: Part 8
Notes

This page intentionally left blank.

© 2020 Sarah Edwards

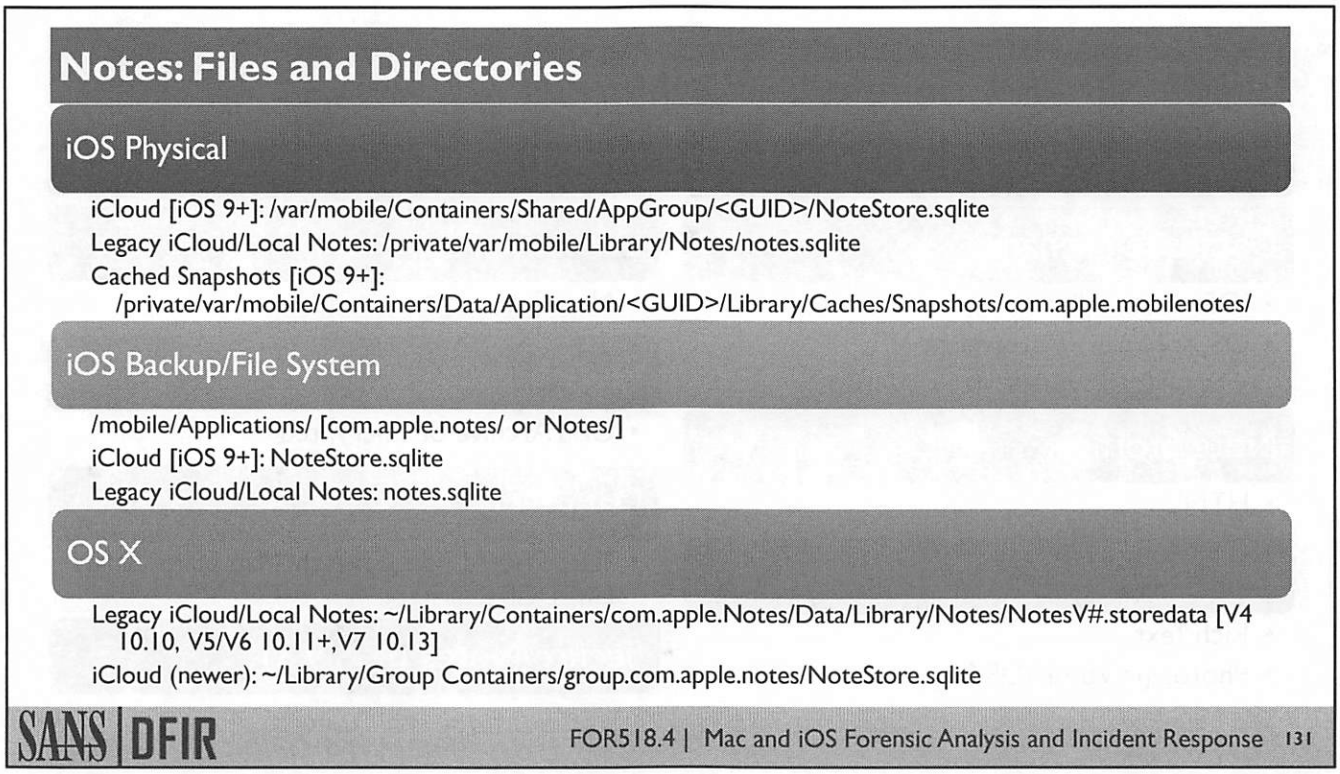


The Notes application allows a user to create notes of various types on macOS, iCloud.com, or on iOS devices. If the user selects one of the syncing services, these notes will be pushed to all devices associated with these accounts.

The iCloud service will sync all iCloud notes to all devices the user has specified to use iCloud services. The user may also choose to create local device notes that do not get synced ("On My Mac").

The Notes application and data files have been rapidly changing ever since it was introduced in 10.8.

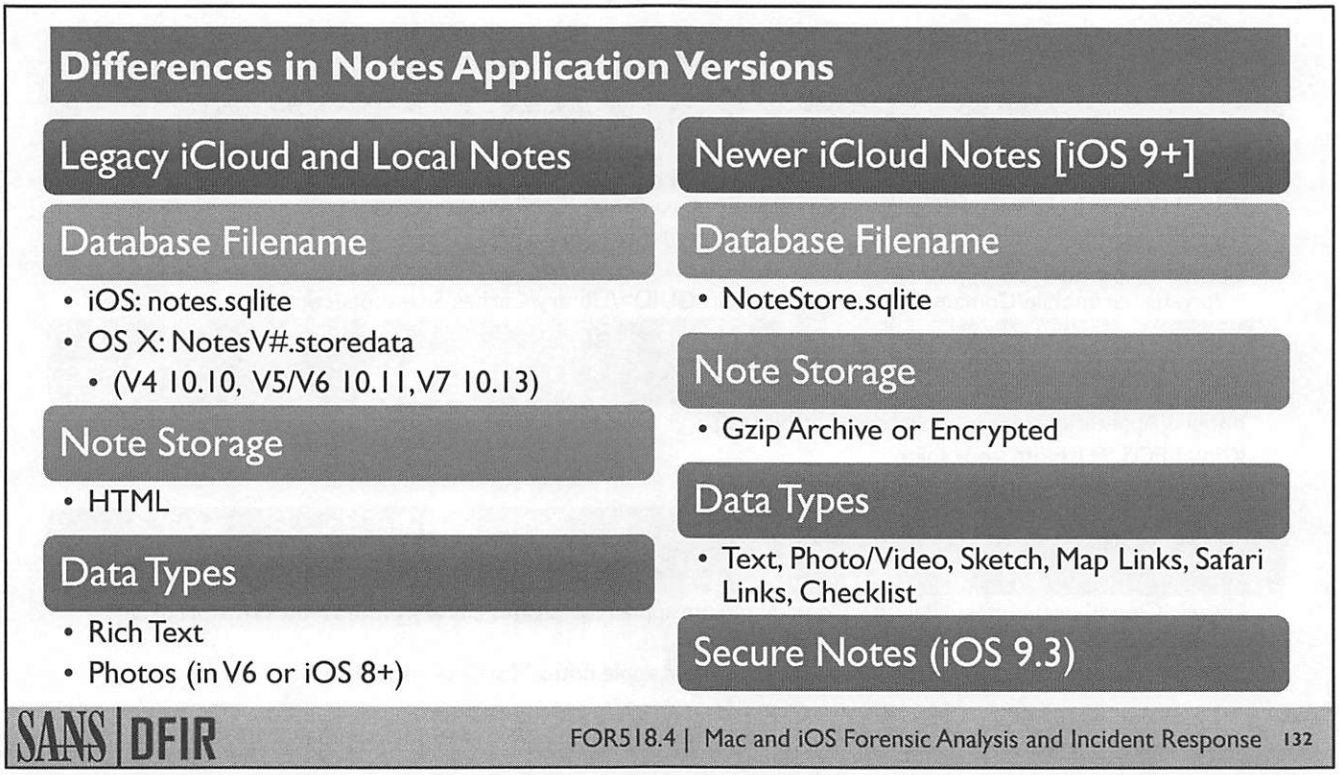
© 2020 Sarah Edwards



The files and paths displayed above show where and how the notes are stored. The Notes application has changed from the `notes.sqlite` database to the `NoteStore.sqlite` database. The introduction of notes that contain images, movies, and other attachments has changed the notes database quite a bit in recent versions.

Attachments:
iOS Physical/File System: `/mobile/Applications/com.apple.notes/Media/<GUID>/`
OS X: `~/Library/Group Containers/group.com.apple.notes/Media/`
OS X: `~/Library/Containers/com.apple.Notes/Data/Library/CoreData/Attachments/`

© 2020 Sarah Edwards



The Legacy iCloud/Local Notes were stored in a simpler database named `notes.sqlite`. On macOS, these databases had a version number associated with them. The newer database file is `NoteStore.sqlite`. Each version of the database may have slight differences in it. The basic note body storage was plaintext HTML, while the newer versions are using Gzip files as a BLOB in a database column.

The newer versions of Notes allow users to embed media such as photo, video, sketches, checklists, and links to Maps, Safari, etc.

Secure notes were introduced with iOS 9.3 and macOS 10.11—in this case, the Gzip data is encrypted. The cryptographic information can be found in various associated columns.

© 2020 Sarah Edwards

Notes Database: notes.sqlite / NotesV#.storedata
Notes and Content

- ZCONTENT: HTML-formatted content
- ZFILENAME: Look in `/attachments` directory for files

```
select
znote.z_pk,
datetime(znote.zcreationdate+978307200,'unixepoch','localtime') as "Creation Date",
datetime(znote.zmodificationdate+978307200,'unixepoch','localtime') as "Modification Date",
znote.zdeletedflag, znote.ztitle,zaccount.zname as "Account Name",znotebody.zcontent,znoteattachment.zfilename
from znote
left join zstore on znote.zstore == zstore.z_pk
left join zaccount on zstore.zaccount == zaccount.z_pk
left join znotebody on znote.zbody == znotebody.z_pk
left join znoteattachment on znote.z_pk == znoteattachment.znote
order by "Creation Date"
```

	Z_PK	Creation Date	Modification Date	ZDELETEDFLAG	ZTITLE	Account Name	ZCONTENT	ZFILENAME
41	828	2016-09-04 17:50:16	2016-09-04 17:50:36	0	Picture Note	sledwards@gmail.com	<html><head></head><body style="word-wrap: break-word; -webkit-nbsp-mode: space; -webkit-line-break: after-white-space;">Picture Note...	IMG_0793.JPG
42	831	2016-09-04 17:50:43	2016-09-04 17:52:16	0	pasword 1	sledwards@gmail.com	<html><head></head><body style="word-wrap: break-word; -webkit-nbsp-mode: space; -webkit-line-break: after-white-space;"> </...	NULL
43	832	2016-09-04 19:59:09	2016-09-04 19:59:27	0	Secrets	sledwards@gmail.com	Secrets Password is skjchBxhd23 	NULL
44	833	2016-09-04 19:59:30	2016-09-04 20:00:23	0	Neat photo	sledwards@gmail.com	Neat photo <object type="application/x-apple-msg-attachment" data="cid:4F975426-267B-4EBA-964B-7622BFB5A5...	unknown.jpg
45	834	2016-09-04 20:10:52	2016-09-04 20:11:54	0	A few of my favorite things:	sledwards@gmail.com	A few of my favorite things: Cats Dogs Cars</div...	NULL

This SQLite query extracts each note item, its timestamps, whether it's marked deleted or not, note title, associated account, HTML content, and attachment filename from the older `notes.sqlite/NotesV#.storedata` files. The attachments can be found in the `/attachments` directory. Look for the matching filename.

The timestamps in this database are stored in Mac Epoch.

```
select
znote.z_pk,
datetime(znote.zcreationdate+978307200,'unixepoch','localtime') as "Creation
Date",
datetime(znote.zmodificationdate+978307200,'unixepoch','localtime') as
"Modification Date",
znote.zdeletedflag,
znote.zsummary,
znote.ztitle,
znote.zauthor,
znote.zguid,
zaccount.zname as "Account Name",
znotebody.zcontent,
znoteattachment.zcontentid,
znoteattachment.zfilename
from znote
left join zstore on znote.zstore == zstore.z_pk
left join zaccount on zstore.zaccount == zaccount.z_pk
left join znotebody on znote.zbody == znotebody.z_pk
left join znoteattachment on znote.z_pk == znoteattachment.znote
order by "Creation Date"
```

© 2020 Sarah Edwards

```
1   select
2   znote.z_pk,
3   datetime(znote.zcreationdate+978307200,'unixepoch','localtime') as "Creation Date",
4   datetime(znote.zmodificationdate+978307200,'unixepoch','localtime') as "Modification Date",
5   znote.zdeletedflag, znote.ztitle,zaccount.zname as "Account Name",znotebody.zcontent,znoteattachment.zfilename
6   from znote
7   left join zstore on znote.zstore == zstore.z_pk
8   left join zaccount on zstore.zaccount == zaccount.z_pk
9   left join znotebody on znote.zbody == znotebody.z_pk
10  left join znoteattachment on znote.z_pk == znoteattachment.znote
11  order by "Creation Date"
```

	Z_PK	Creation Date	Modification Date	ZDELETEDFLAG	ZTITLE	Account Name	ZCONTENT	ZFILENAME
41	828	2016-09-04 17:50:16	2016-09-04 17:50:36	0	Picture Note	sledwards@gmail.com	<html><head></head><body style="word-wrap: break-word; -webkit-nbsp-mode: space; -webkit-line-break: after-white-space;">Picture Note…	IMG_0793.JPG
42	831	2016-09-04 17:50:43	2016-09-04 17:52:16	0	pasword 1	sledwards@gmail.com	<html><head></head><body style="word-wrap: break-word; -webkit-nbsp-mode: space; -webkit-line-break: after-white-space;"> </…	NULL
43	832	2016-09-04 19:59:09	2016-09-04 19:59:27	0	Secrets	sledwards@gmail.com	Secrets Password is skjchBxhd23 	NULL
44	833	2016-09-04 19:59:30	2016-09-04 20:00:23	0	Neat photo	sledwards@gmail.com	Neat photo <object type="application/x-apple-msg-attachment" data="cid:4F975426-267B-4EBA-964B-7622BFB5A5…	unknown.jpg
45	834	2016-09-04 20:10:52	2016-09-04 20:11:54	0	A few of my favorite things:	sledwards@gmail.com	A few of my favorite things: Cats Dogs Cars</div…	NULL

© 2020 Sarah Edwards

Notes Database [iOS 9+]: NotesStore.sqlite
Note Folders

- Folders:
- ZICCLOUDSYNCINGOBJECT.Z_ENT == 12


```
1   select
2   ZICCLOUDSYNCINGOBJECT.Z_PK,
3   ZICCLOUDSYNCINGOBJECT.zaccountnameforaccountlistsorting as "Account Name",
4   ZICCLOUDSYNCINGOBJECT.zidentifier,
5   ZICCLOUDSYNCINGOBJECT.ztitle2 as "Folder Title",
6   ZICCLOUDSYNCINGOBJECT.zdateforlasttitlemodification,
7   datetime(ZICCLOUDSYNCINGOBJECT.zdateforlasttitlemodification+978307200,'unixepoch','localtime') as "Folder Title Last Modified",
8   ZICCLOUDSYNCINGOBJECT.zmarkedfordeletion
9   from ZICCLOUDSYNCINGOBJECT
10  where ZICCLOUDSYNCINGOBJECT.z_ent == 12 --Note Folders are Z_ENT = 12
```

	Z_PK	Account Name	ZIDENTIFIER	Folder Title	ZDATEFORLASTTITLEMODIFICATION	Folder Title Last Modified	ZMARKEDFORDELETION
1	2	1_iCloud	DefaultFolder-CloudKit	Notes	NULL	NULL	0
2	3	1_iCloud	TrashFolder-CloudKit	Recently Deleted	NULL	NULL	0
3	66	1_iCloud	336F4A56-18D0-421A-AA9A-DAAA3C1C9D1C	Research	469407828.849662	2015-11-16 18:03:48	0
4	104	1_iCloud	EBE5D44E-310E-4018-8F41-7842C3676851	My Secrets	493489715.001946	2016-08-21 12:28:35	0

135

In the newer Notes SQLite database, this query extracts each note folder, associated account name, when the folder was last modified, and whether it's marked for deletion.

The "Z_ENT" 12 is selected here to gather folder data. 12 is the key for "ICFolder" information. The screenshot to the right shows other data that may be extracted.

The Last Modified timestamp is stored as Mac Epoch.

```
select
ZICCLOUDSYNCINGOBJECT.Z_PK,
ZICCLOUDSYNCINGOBJECT.zaccountnameforaccountlistsorting
as "Account Name",
ZICCLOUDSYNCINGOBJECT.zidentifier,
ZICCLOUDSYNCINGOBJECT.ztitle2 as "Folder Title",
ZICCLOUDSYNCINGOBJECT.zdateforlasttitlemodification,dat
etime(ZICCLOUDSYNCINGOBJECT.zdateforlasttitlemodificati
on+978307200,'unixepoch','localtime') as "Folder Title
Last Modified",
ZICCLOUDSYNCINGOBJECT.zmarkedfordeletionfrom
ZICCLOUDSYNCINGOBJECT
where ZICCLOUDSYNCINGOBJECT.z_ent == 12 are Z_ENT = 12
```

Table: Z_PRIMARYKEY

	Z_ENT	Z_NAME
	Filter	Filter
1	1	ICAuthor
2	2	ICCloudState
3	3	ICCloudSyncingObject
4	4	ICAttachment
5	5	ICAttachmentPreviewImage
6	6	ICDeviceMigrationState
7	7	ICLegacyTombstone
8	8	ICMedia
9	9	ICNote
10	10	ICNoteContainer
11	11	ICAccount
12	12	ICFolder
13	13	ICDDevice
14	14	ICGroup
15	15	ICLocation
16	16	ICAttachmentLocation
17	17	ICNoteChange
18	18	ICNoteData
19	19	ICPerson
20	20	ICSearchIndexTransaction
21	21	ICServerChangeToken
22	22	NextId

© 2020 Sarah Edwards

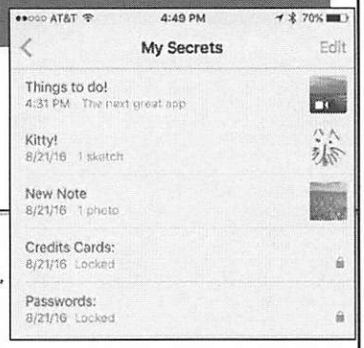
```
select
ZICCLOUDSYNCINGOBJECT.Z_PK,
ZICCLOUDSYNCINGOBJECT.zaccountnameforaccountlistsorting as "Account Name",
ZICCLOUDSYNCINGOBJECT.zidentifier,
ZICCLOUDSYNCINGOBJECT.ztitle2 as "Folder Title",
ZICCLOUDSYNCINGOBJECT.zdateforlasttitlemodification,
datetime(ZICCLOUDSYNCINGOBJECT.zdateforlasttitlemodification+978307200,'unixepoch','localtime') as "Folder Title Last Modified",
ZICCLOUDSYNCINGOBJECT.zmarkedfordeletion
from ZICCLOUDSYNCINGOBJECT
where ZICCLOUDSYNCINGOBJECT.z_ent == 12 --Note Folders are Z_ENT = 12
```

	Z_PK	Account Name	ZIDENTIFIER	Folder Title	ZDATEFORLASTTITLEMODIFICATION	Folder Title Last Modified	ZMARKEDFORDELETION
1	2	1_iCloud	DefaultFolder-CloudKit	Notes	*NULL*	*NULL*	0
2	3	1_iCloud	TrashFolder-CloudKit	Recently Deleted	*NULL*	*NULL*	0
3	66	1_iCloud	336F4A56-18D0-421A-AA9A-DAAA3C1C9D1C	Research	469407828.849662	2015-11-16 18:03:48	0
4	104	1_iCloud	EBE5D44E-310E-4018-8F41-7842C3676851	My Secrets	493489715.001946	2016-08-21 12:28:35	0

© 2020 Sarah Edwards

Notes Database [iOS 9+]: NotesStore.sqlite Notes

- Notes under "My Secrets" Folder (#104)
- Secure Notes: Password-Protected Notes
- ZDATA = Note Contents




```
2  ZICCLOUDSYNCINGOBJECT.Z_PK,ZICCLOUDSYNCINGOBJECT.zidentifier,Z_12NOTES.Z_12FOLDERS as "Note Folder",
3  datetime(ZICCLOUDSYNCINGOBJECT.zcreationdate+978307200,'unixepoch','localtime') as "Create Time",
4  datetime(ZICCLOUDSYNCINGOBJECT.zmodificationdate1+978307200,'unixepoch','localtime') as "Modification Time",
5  datetime(ZICCLOUDSYNCINGOBJECT.zfoldersmodificationdate+978307200,'unixepoch','localtime') as "Folder Modification Time",
6  ZICCLOUDSYNCINGOBJECT.ztitle1 as "Note Title",ZICCLOUDSYNCINGOBJECT.zsnippet,ZICCLOUDSYNCINGOBJECT.zispasswordprotected as "Protected",
7  ZICCLOUDSYNCINGOBJECT.zpasswordhint,ZICCLOUDSYNCINGOBJECT.zmarkedfordeletion as "Deleted",ZICNOTEDATA.ZDATA
8  from ZICCLOUDSYNCINGOBJECT
9  left join ZICNOTEDATA on ZICCLOUDSYNCINGOBJECT.znotedata == ZICNOTEDATA.Z_pk
10 left join Z_12NOTES on ZICCLOUDSYNCINGOBJECT.z_pk == Z_12NOTES.Z_9NOTES
11 where ZICCLOUDSYNCINGOBJECT.z_ent == 9 and "Note Folder" == 104
12 order by "Create Time"
```

	Z_PK	ZIDENTIFIER	Note Folder	Create Time	Modification Time	Folder Modification Time	Note Title	ZSNIPPET	Protected	ZPASSWORDHINT	Deleted	ZDATA
1	136	3E8C7FCC-A1F2-4641-8320-41F5EA81B0B7	104	2016-08-21 12:28:37	2016-08-21 12:29:33	2016-08-21 13:04:33	Passwords:	NULL	1	it's password	0	BLOB
2	135	5DD1257F-EE19-42CF-9A25-89CE7575D203	104	2016-08-21 12:29:39	2016-08-21 12:31:19	2016-08-21 13:04:17	Credits Cards:	NULL	1	it's password	0	...
3	107	B0F078A1-855B-43DE-B945-7FB075906DF1	104	2016-08-21 12:32:41	2016-08-21 12:33:52		New Note		0		0	
4	112	23511203-24B4-47F4-8066-281864C5CE0B	104	2016-08-21 12:34:01	2016-08-21 12:34:20	NULL	New Note		0	NULL	0	BLOB
5	119	76B2B660-E815-43D7-BFF2-A4921C02059C	104	2016-08-21 12:34:26	2016-08-21 18:39:19	NULL	Things to do!	The next great app	0	NULL	0	BLOB
6	126	76B0A953-0B2C-4FC7-8DD0-D0F7EA69841D	104	2016-08-21 12:36:42	2016-08-21 12:37:43	NULL	Kitty!		0	NULL	0	BLOB

This query extracts the Note data to include timestamps, title, snippets, and whether or not it is encrypted and/or marked for deletion. The highlighted row contains a note that is named "`New Note`" and is not encrypted. If it was, the Protected column would show a "`1`". The note body is stored in a GZIP archive in the ZDATA column in the BLOB.

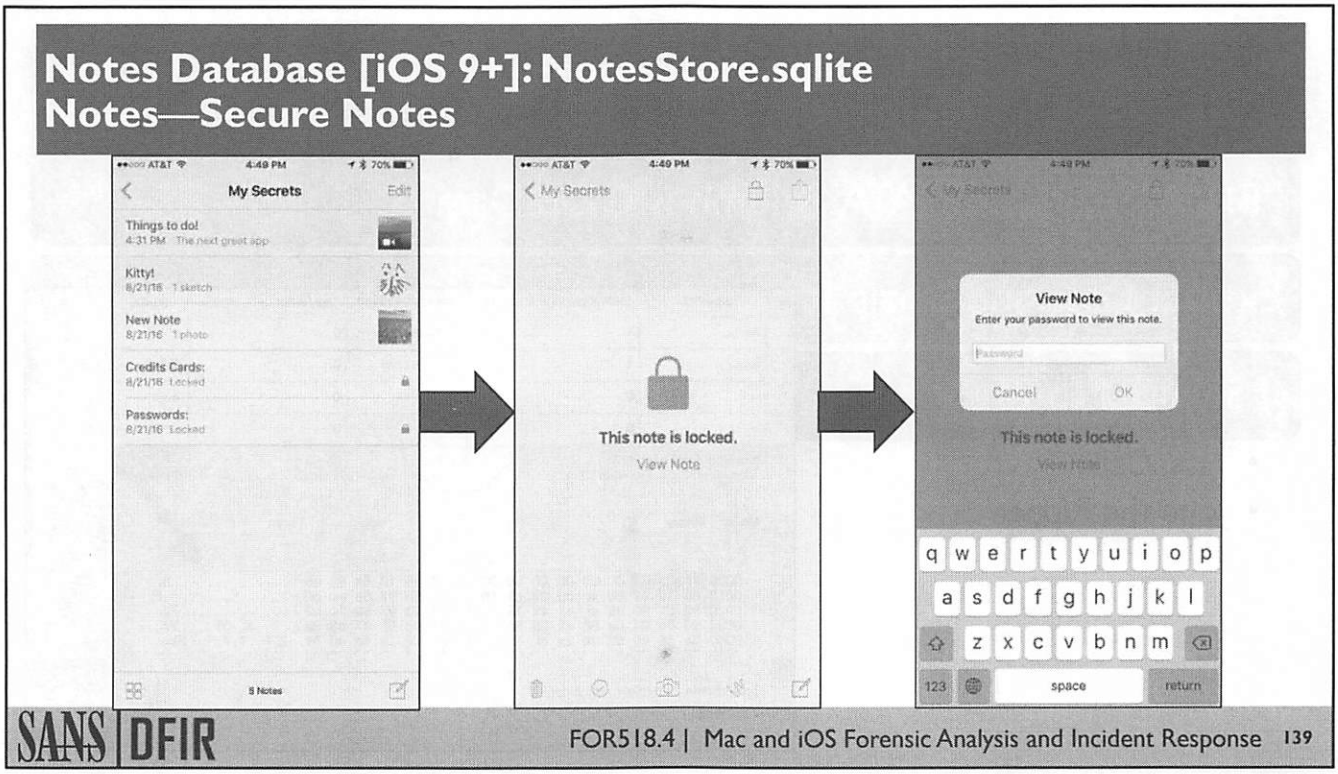
```
select
ZICCLOUDSYNCINGOBJECT.Z_PK,
ZICCLOUDSYNCINGOBJECT.zidentifier,
Z_12NOTES.Z_12FOLDERS as "Note Folder",
datetime(ZICCLOUDSYNCINGOBJECT.zcreationdate+978307200,'unixepoch','localtime
') as "Create Time",
datetime(ZICCLOUDSYNCINGOBJECT.zmodificationdate1+978307200,'unixepoch','loca
ltime') as "Modification Time",
datetime(ZICCLOUDSYNCINGOBJECT.zfoldersmodificationdate+978307200,'unixepoch'
,'localtime') as "Folder Modification Time",
ZICCLOUDSYNCINGOBJECT.ztitle1 as "Note Title",
ZICCLOUDSYNCINGOBJECT.zsnippet,
ZICCLOUDSYNCINGOBJECT.zispasswordprotected as "Protected",
ZICCLOUDSYNCINGOBJECT.zpasswordhint,
ZICCLOUDSYNCINGOBJECT.zmarkedfordeletion as "Deleted",
ZICNOTEDATA.ZDATA from ZICCLOUDSYNCINGOBJECT
left join ZICNOTEDATA on ZICCLOUDSYNCINGOBJECT.znotedata == ZICNOTEDATA.Z_pk
left join Z_12NOTES on ZICCLOUDSYNCINGOBJECT.z_pk == Z_12NOTES.Z_9NOTES
where ZICCLOUDSYNCINGOBJECT.z_ent == 9 and "Note Folder" == 104
order by "Create Time"
```

© 2020 Sarah Edwards

```
2   ZICCLOUDSYNCINGOBJECT.Z_PK,ZICCLOUDSYNCINGOBJECT.zidentifier,Z_12NOTES.Z_12FOLDERS as "Note Folder",
3   datetime(ZICCLOUDSYNCINGOBJECT.zcreationdate+978307200,'unixepoch','localtime') as "Create Time",
4   datetime(ZICCLOUDSYNCINGOBJECT.zmodificationdate1+978307200,'unixepoch','localtime') as "Modification Time",
5   datetime(ZICCLOUDSYNCINGOBJECT.zfoldersmodificationdate+978307200,'unixepoch','localtime') as "Folder Modification Time",
6   ZICCLOUDSYNCINGOBJECT.ztitle1 as "Note Title",ZICCLOUDSYNCINGOBJECT.zsnippet,ZICCLOUDSYNCINGOBJECT.zispasswordprotected as "Protected",
7   ZICCLOUDSYNCINGOBJECT.zpasswordhint,ZICCLOUDSYNCINGOBJECT.zmarkedfordeletion as "Deleted",ZICNOTEDATA.ZDATA
8   from ZICCLOUDSYNCINGOBJECT
9   left join ZICNOTEDATA on ZICCLOUDSYNCINGOBJECT.znotedata == ZICNOTEDATA.Z_pk
10  left join Z_12NOTES on ZICCLOUDSYNCINGOBJECT.z_pk == Z_12NOTES.Z_9NOTES
11  where ZICCLOUDSYNCINGOBJECT.z_ent == 9 and "Note Folder" == 104
12  order by "Create Time"
```

	Z_PK	ZIDENTIFIER	Note Folder	Create Time	Modification Time	Folder Modification Time	Note Title	ZSNIPPET	Protected	ZPASSWORDHINT	Deleted	ZDATA
1	136	3E8C7FCC-A1F2-4641-8320-41F5EA81B0B7	104	2016-08-21 12:28:37	2016-08-21 12:29:33	2016-08-21 13:04:33	Passwords:	NULL	1	it's password	0	BLOB
2	135	5DD1257F-EE19-42CF-9A25-89CE7575D203	104	2016-08-21 12:29:39	2016-08-21 12:31:19	2016-08-21 13:04:17	Credits Cards:	NULL	1	it's password	0	...
3	107	B0F078A1-855B-43DE-B945-7F60759D6DF1	104	2016-08-21 12:32:41	2016-08-21 12:33:52	NULL	New Note		0	NULL	0	
4	112	23511203-24B4-47F4-8066-281864C5CE0B	104	2016-08-21 12:34:01	2016-08-21 12:34:20	NULL	New Note		0	NULL	0	BLOB
5	119	76B2B660-E815-43D7-BFF2-A4921C02059C	104	2016-08-21 12:34:26	2016-08-21 18:39:19	NULL	Things to do!	The next great app	0	NULL	0	BLOB
6	126	76B0A953-0B2C-4FC7-8DD0-D0F7EA69841D	104	2016-08-21 12:36:42	2016-08-21 12:37:43	NULL	Kitty!		0	NULL	0	BLOB

© 2020 Sarah Edwards



With iOS 9.3 and macOS 10.11, the secure notes feature was introduced. Secure notes may contain sensitive data, pictures, and/or other media. The example above shows the "`My Secrets`" note folder, which contains a couple of Secure Notes. These notes are marked by the small lock icon.

If a secure note is selected, the user will get a "`This note is locked.`" message and will have to input their Notes password to view the data.

Secure Notes is only supported for iCloud notes; it is not available for local device notes.

© 2020 Sarah Edwards

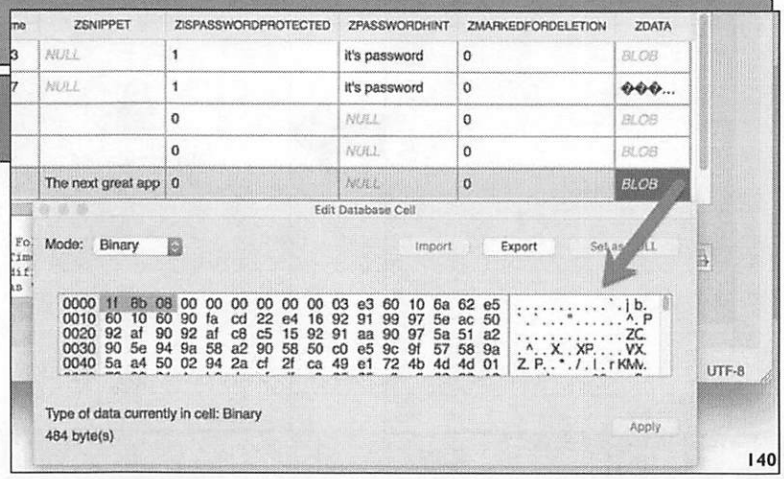
Notes Database [iOS 9+]: NotesStore.sqlite
Notes—Note Contents [1]

"BLOB" data in ZDATA column is GZip Data

Extract Data and Unzip It

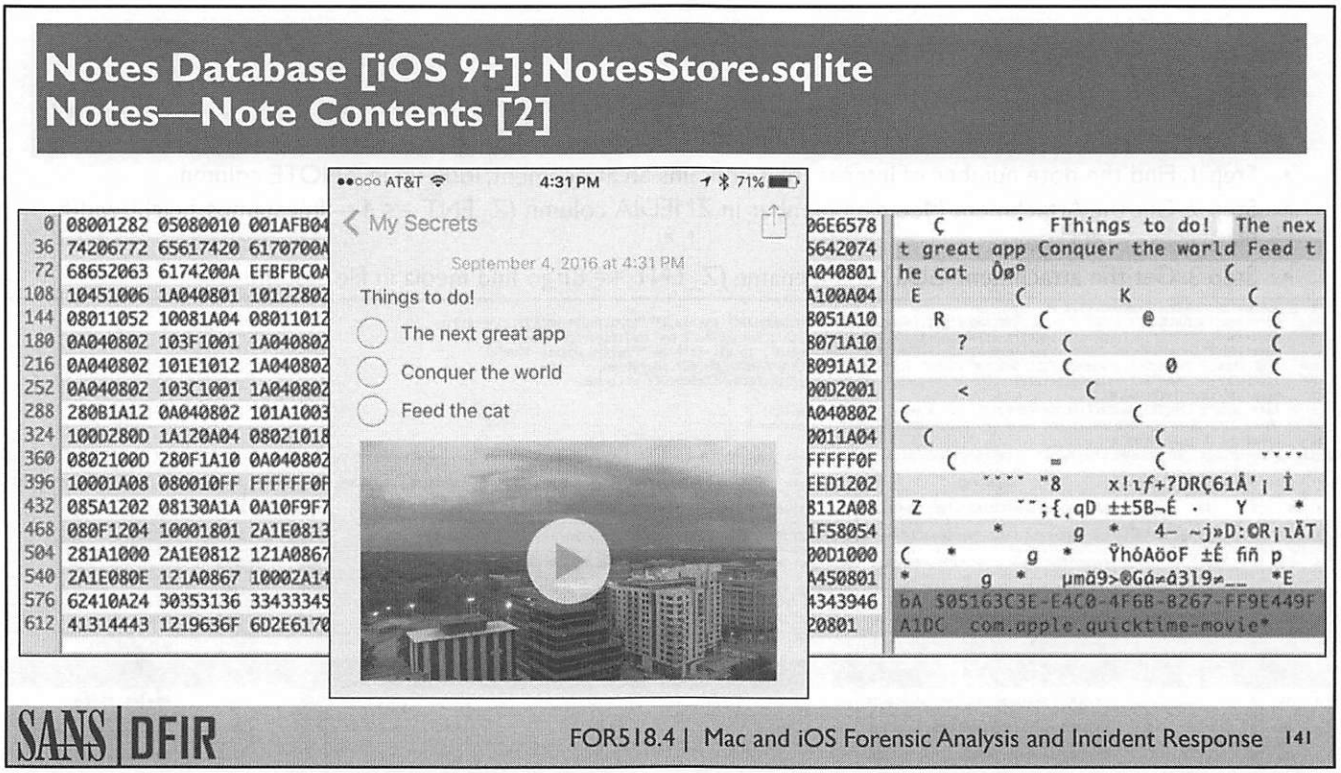
If Secure Note:

- HashCat
- John the Ripper



Assuming the note is not encrypted, the user can extract the note data by viewing the BLOB data in the `ZDATA` column. The screenshot shows the SQLite Database Browser opening the BLOB in a hex viewer. The highlighted file header "`1f 8b 08`" is the file signature for Gzip data. This archive can be saved to disk and opened with any GZIP compatible software.

© 2020 Sarah Edwards



The example note contains a list of items and a movie.

Once unarchived, the note data will have a proprietary format, as shown above. The highlighted sections show the checklist; however, it may be hard to determine which items are checklist titles and which are checklist items.

The movie can be reviewed by taking note of the GUID next to the "`com.apple.quicktime-movie`" file-type string.

© 2020 Sarah Edwards

Notes Database [iOS 9+]: NotesStore.sqlite
Note Attachments/Media—In Database

- Step 1: Find the note number of interest that contains an attachment; look up in ZNOTE column
- Step 2: Get the Attachment Metadata number in ZMEDIA column (Z_ENT == 4—timestamps, height/width, file size, type, duration)
- Step 3: Get the attachment GUID and filename (Z_ENT == 8); go find media in file system

```
1  select ZICCLOUDSYNCINGOBJECT.Z_PK,
2  ZICCLOUDSYNCINGOBJECT.Z_ENT,ZICCLOUDSYNCINGOBJECT.znote,ZICCLOUDSYNCINGOBJECT.zmedia,ZICCLOUDSYNCINGOBJECT.zidentifier,
3  datetime(ZICCLOUDSYNCINGOBJECT.zmodificationdate+978307200,'unixepoch','localtime') as "Modification Time",
4  datetime(ZICCLOUDSYNCINGOBJECT.zpreviewupdatedate+978307200,'unixepoch','localtime') as "Preview Update Time",
5  ZICCLOUDSYNCINGOBJECT.ztypeuti,ZICCLOUDSYNCINGOBJECT.zfilesize,ZICCLOUDSYNCINGOBJECT.zduration,
6  ZICCLOUDSYNCINGOBJECT.zsizeheight,ZICCLOUDSYNCINGOBJECT.zsizewidth,ZICCLOUDSYNCINGOBJECT.zfilename
7  from ZICCLOUDSYNCINGOBJECT
8  left join Z_12NOTES on ZICCLOUDSYNCINGOBJECT.z_pk == Z_12NOTES.Z_9NOTES
9  where ZICCLOUDSYNCINGOBJECT.z_ent == 4 or ZICCLOUDSYNCINGOBJECT.z_ent == 8
10 order by "Create Time"
```

	Z_PK	Z_ENT	ZNOTE	ZMEDIA	ZIDENTIFIER	Modification Time	Preview Update Time	ZTYPEUTI	ZFILESIZE	ZDURATION	ZSIZEHEIGHT	ZSIZEWIDTH	ZFILENAME
1	79	4	78	80	8A0976B7-4A8B-4CDA-A001-366171E42C2B	2016-02-11 16:09:34	2016-02-11 16:09:35	public.jpeg	1872097	0.0	3264.0	2448.0	NULL
2	108	4	107 (1)	109 (2)	C6EEEFDB-F5F7-4F40-8321-D1A5D0DAE153	2016-08-21 12:33:51	2016-08-21 12:33:52	public.jpeg	3380346	0.0	3264.0	2448.0	NULL
3	113	4	112	114	FA020FE0-D089-4C82-8F41-096DBE1AB583	2016-08-21 12:34:19	2016-08-21 12:34:19	com.apple.notes.sketch	0	0.0	566.4	629.6	NULL
4	122	4	119	123	05163C3E-E4C0-4F6B-B267-FF9E449FA1DC	2016-08-21 12:36:30	2016-08-21 12:36:31	com.apple.quicktime-movie	24409651	40.06	720.0	1280.0	NULL
5	127	4	126	128	EEE54145-B653-49B2-94CE-7CEF6A63AE17	2016-08-21 12:37:17	2016-08-21 12:37:17	com.apple.notes.sketch	0	0.0	871.2	744.8	NULL
6	80 (3)	8	NULL	NULL	48299DC6-97A3-4C0F-A63B-BCCDDC7F44E0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	IMG_0132.JPG
7	109	8	NULL	NULL	1E115980-2711-4604-A36C-6F53836F601	NULL	NULL	NULL	NULL	NULL	NULL	NULL	IMG_0438.JPG
8	114	8	NULL	NULL	499BDC14-CF9C-45C9-9D60-6D83A9F208C4	NULL	NULL	NULL	NULL	NULL	NULL	NULL	New Note.drawing
9	123	8	NULL	NULL	F1034C91-83E4-42BD-B543-47D93C5DC4F3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	trim.B6A8F9CF-7EFD-4ABC-BF89-71A3CF8B1237.MOV
10	128	8	NULL	NULL	C5F29B15-BBEA-41B6-A53B-00A6B2310BF3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	New Note.drawing

To associate an attachment (media) with an actual file, this query will have to be performed. This query extracts the note number, media number, GUID, timestamps, file type, file metadata, and filename. The numbered steps above can get you to associate a certain file's metadata with the actual filename.

Other types of "`attachments`" like URLs and location data may also want to be extracted. Examples of these are shown below:

ZSUMMARY	ZTITLE	ZTYPEUTI	ZURLSTRING
Filter	Filter	public.url	Filter
NULL	The White House	public.url	https://maps.apple.com/maps?address=1600%20Pennsylvania%2...
See the President's daily schedule, explore behind-the-s...	The White House	public.url	https://www.whitehouse.gov/
	Find a Shop - Stewart's ShopsStewart's Shops	public.url	http://www.stewartsshops.com/find-a-shop-2/

Table: ZICLOCATION

	Z_PK	Z_ENT	Z_OPT	ZPLACEUPDATEI	ZATTACHMENT	ZLATITUDE	ZLONGITUDE	ZPLACEMARK
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	16	1	1	183	38.897517	-77.036542	BLOB

© 2020 Sarah Edwards

```
select
ZICCLOUDSYNCINGOBJECT.Z_PK,
ZICCLOUDSYNCINGOBJECT.Z_ENT,
ZICCLOUDSYNCINGOBJECT.znote,
ZICCLOUDSYNCINGOBJECT.zmedia,
ZICCLOUDSYNCINGOBJECT.zidentifier,
datetime(ZICCLOUDSYNCINGOBJECT.zmodificationdate+978307200,'unixepoch','local
time') as "Modification Time",
datetime(ZICCLOUDSYNCINGOBJECT.zpreviewupdatedate+978307200,'unixepoch','loca
ltime') as "Preview Update Time",
ZICCLOUDSYNCINGOBJECT.ztypeuti,
ZICCLOUDSYNCINGOBJECT.zfilesize,
ZICCLOUDSYNCINGOBJECT.zduration,
ZICCLOUDSYNCINGOBJECT.zsizeheight,
ZICCLOUDSYNCINGOBJECT.zsizewidth,
ZICCLOUDSYNCINGOBJECT.zfilename
from ZICCLOUDSYNCINGOBJECTleft join Z_12NOTES
on ZICCLOUDSYNCINGOBJECT.z_pk == Z_12NOTES.Z_9NOTES
where ZICCLOUDSYNCINGOBJECT.z_ent == 4 or ZICCLOUDSYNCINGOBJECT.z_ent == 8
order by "Create Time"
```

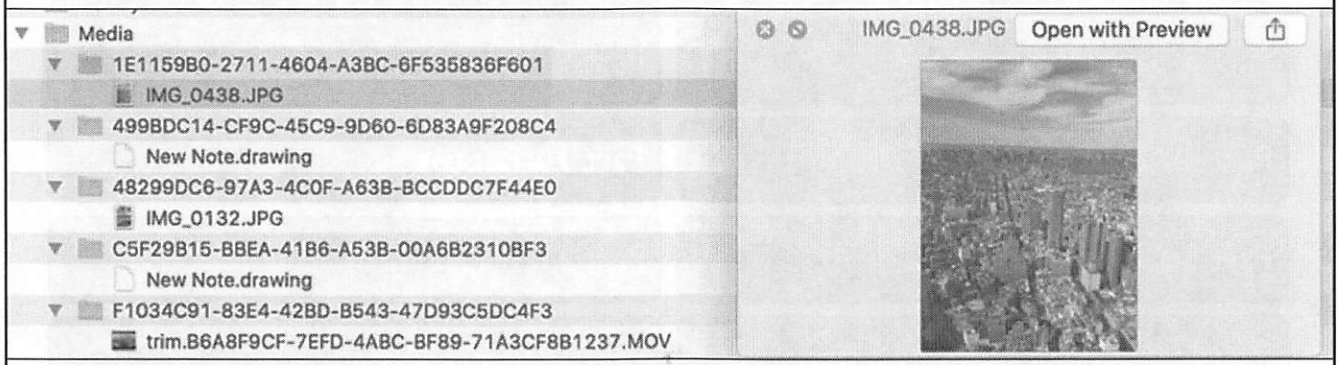
```
select ZICCLOUDSYNCINGOBJECT.Z_PK,
ZICCLOUDSYNCINGOBJECT.Z_ENT,ZICCLOUDSYNCINGOBJECT.znote,ZICCLOUDSYNCINGOBJECT.zmedia,ZICCLOUDSYNCINGOBJECT.zidentifier,
datetime(ZICCLOUDSYNCINGOBJECT.zmodificationdate+978307200,'unixepoch','localtime') as "Modification Time",
datetime(ZICCLOUDSYNCINGOBJECT.zpreviewupdatedate+978307200,'unixepoch','localtime') as "Preview Update Time",
ZICCLOUDSYNCINGOBJECT.ztypeuti,ZICCLOUDSYNCINGOBJECT.zfilesize,ZICCLOUDSYNCINGOBJECT.zduration,
ZICCLOUDSYNCINGOBJECT.zsizeheight,ZICCLOUDSYNCINGOBJECT.zsizewidth,ZICCLOUDSYNCINGOBJECT.zfilename
from ZICCLOUDSYNCINGOBJECT
left join Z_12NOTES on ZICCLOUDSYNCINGOBJECT.z_pk == Z_12NOTES.Z_9NOTES
where ZICCLOUDSYNCINGOBJECT.z_ent == 4 or ZICCLOUDSYNCINGOBJECT.z_ent == 8
order by "Create Time"
```

	Z_PK	Z_ENT	ZNOTE	ZMEDIA	ZIDENTIFIER	Modification Time	Preview Update Time	ZTYPEUTI	ZFILESIZE	ZDURATION	ZSIZEHEIGHT	ZSIZEWIDTH	ZFILENAME
1	79	4	78	80	8A0976B7-4A8B-4CDA-A001-366171E42C2B	2016-02-11 16:09:34	2016-02-11 16:09:35	public.jpeg	1872097	0.0	3264.0	2448.0	NULL
2	108	4	107 (1)	109 (2)	C6EEEFDB-F5F7-4F40-8321-D1A5D0DAE153	2016-08-21 12:33:51	2016-08-21 12:33:52	public.jpeg	3380346	0.0	3264.0	2448.0	NULL
3	113	4	112	114	FA020FE0-D069-4C82-8F41-096DBE1AB583	2016-08-21 12:34:19	2016-08-21 12:34:19	com.apple.notes.sketch	0	0.0	566.4	629.6	NULL
4	122	4	119	123	05163C3E-E4C0-4F6B-B267-FF9E449FA1DC	2016-08-21 12:36:30	2016-08-21 12:36:31	com.apple.quicktime-movie	24409651	40.06	720.0	1280.0	NULL
5	127	4	126	128	EEE54145-B653-49B2-94CE-7CEF6A63AE17	2016-08-21 12:37:17	2016-08-21 12:37:17	com.apple.notes.sketch	0	0.0	871.2	744.8	NULL
6	80 (3)	8	NULL	NULL	48299DC6-97A3-4C0F-A63B-BCCDDC7F44E0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	IMG_0132.JPG
7	109	8	NULL	NULL	1E1159B0-2711-4604-A3BC-6F535836F601	NULL	NULL	NULL	NULL	NULL	NULL	NULL	IMG_0438.JPG
8	114	8	NULL	NULL	499BDC14-CF9C-45C9-9D60-6D83A9F208C4	NULL	NULL	NULL	NULL	NULL	NULL	NULL	New Note.drawing
9	123	8	NULL	NULL	F1034C91-83E4-42BD-B543-47D93C5DC4F3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	trim.B6A8F9CF-7EFD-4ABC-BF89-71A3CF8B1237.MOV
10	128	8	NULL	NULL	C5F29B15-BBEA-41B6-A53B-00A6B2310BF3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	New Note.drawing

© 2020 Sarah Edwards

Notes Database [iOS 9+]: NotesStore.sqlite
Note Attachments/Media—In File System

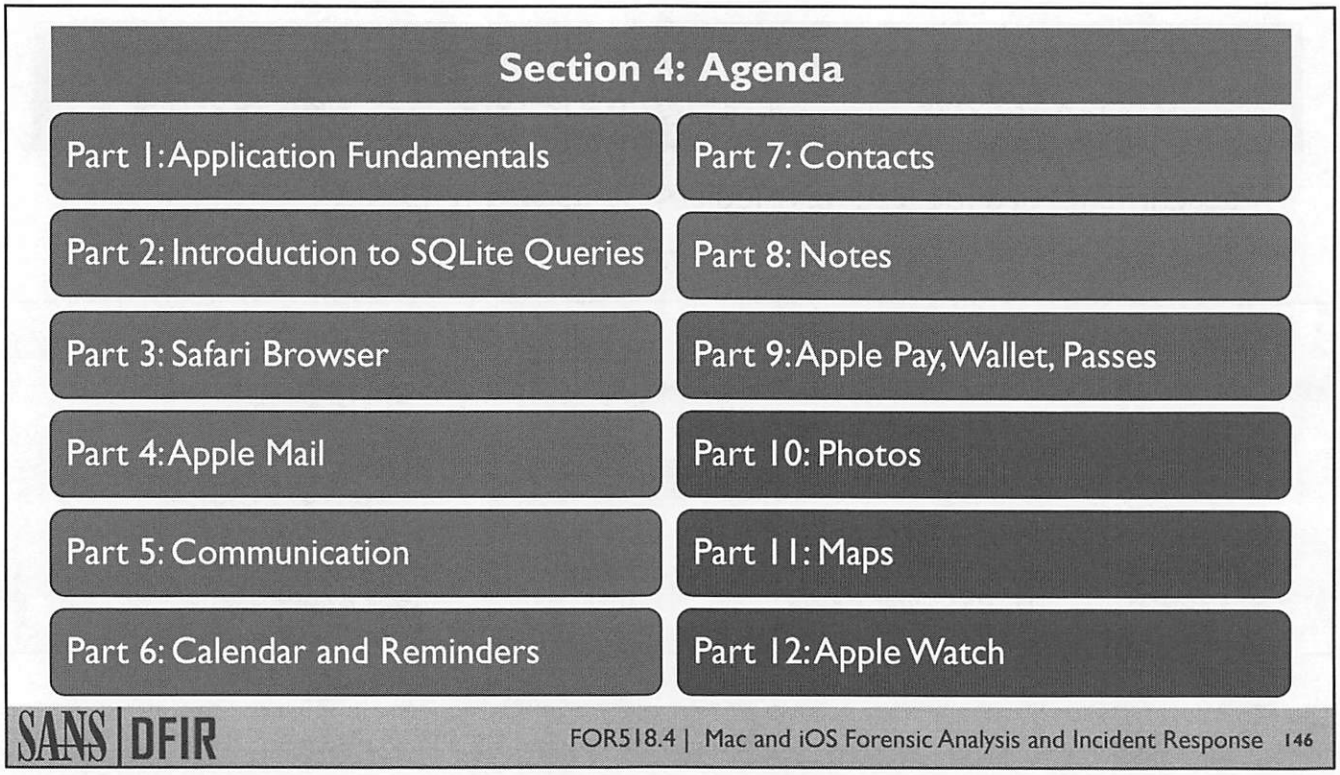
- Attachments are located in: /Media/<GUID_from_DB>/
- Filenames match those in database correlated with note/media rows



Attachments:
iOS Physical/File System: `<notesdir>/Media` or `/attachments`
OS X: `~/Library/Group Containers/group.com.apple.notes/Media/`
OS X: `~/Library/Containers/com.apple.Notes/Data/Library/CoreData/Attachments/`

Once the attachment GUIDs are identified, the analyst can look for the GUID in the `/Attachments` or `/Media` file paths and find the attached file.

© 2020 Sarah Edwards



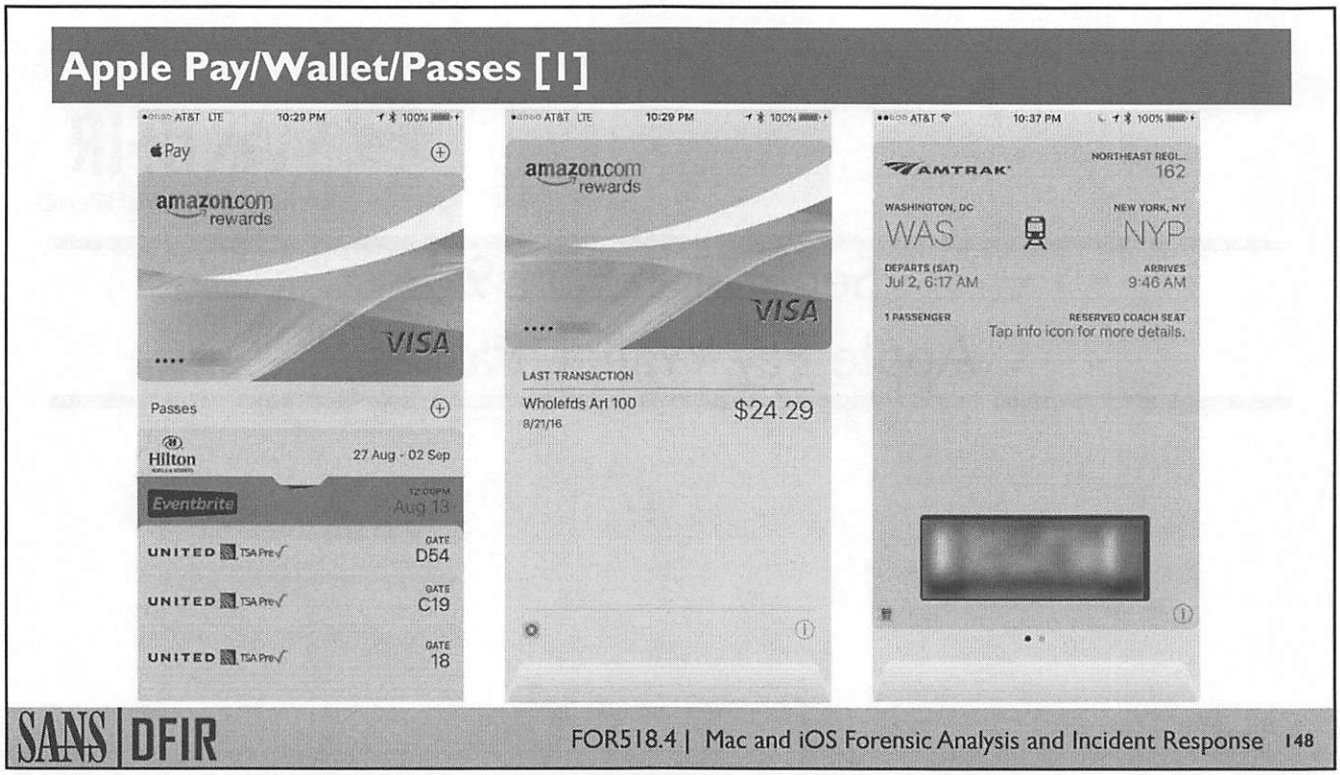
This page intentionally left blank.

© 2020 Sarah Edwards

Section 4: Part 9
Apple Pay, Wallet, Passes

This page intentionally left blank.

© 2020 Sarah Edwards



The Passes application, now called Wallet, can be used to keep track of various tickets and cards. If the user selects to add a credit card to the Apple Pay portion of this application, it can be used to purchase items as well, as shown in the middle screenshot.

© 2020 Sarah Edwards

Apple Pay/Wallet/Passes [2]

Apple Pay

- Credit Cards, Potential Transaction History

Passes

- Tickets (Airline, Train, Movie, Events), Gift Cards, Reservations

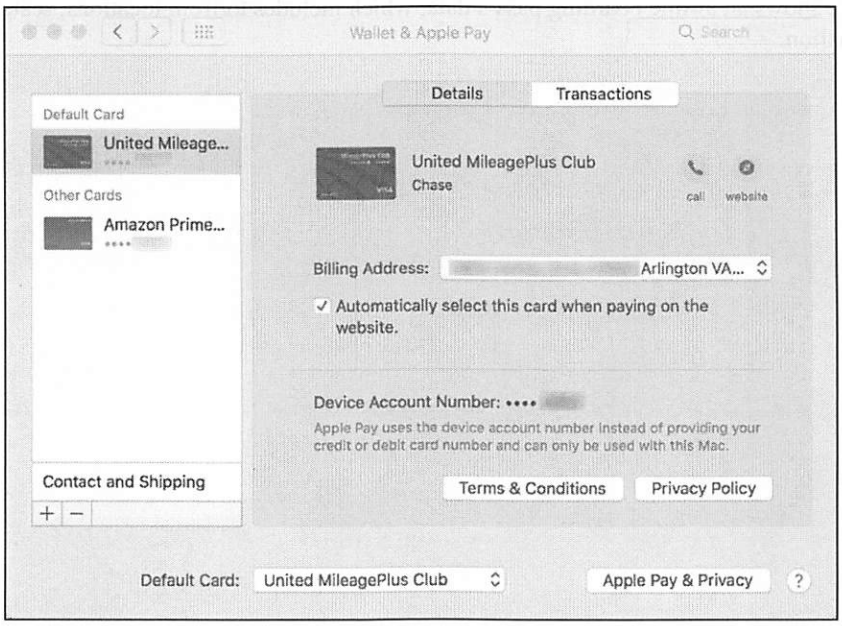
File/Directory Locations

- iOS: May be duplicative
  - /private/var/mobile/Library/Passes/Cards/
  - (iCloud) /private/var/mobile/Library/Mobile Documents/com~apple~shoebox/UbiquitousCards/
- macOS (iCloud): ~/Library/Mobile Documents/com~apple~shoebox/UbiquitousCards/
- macOS: ~/Library/Passes/

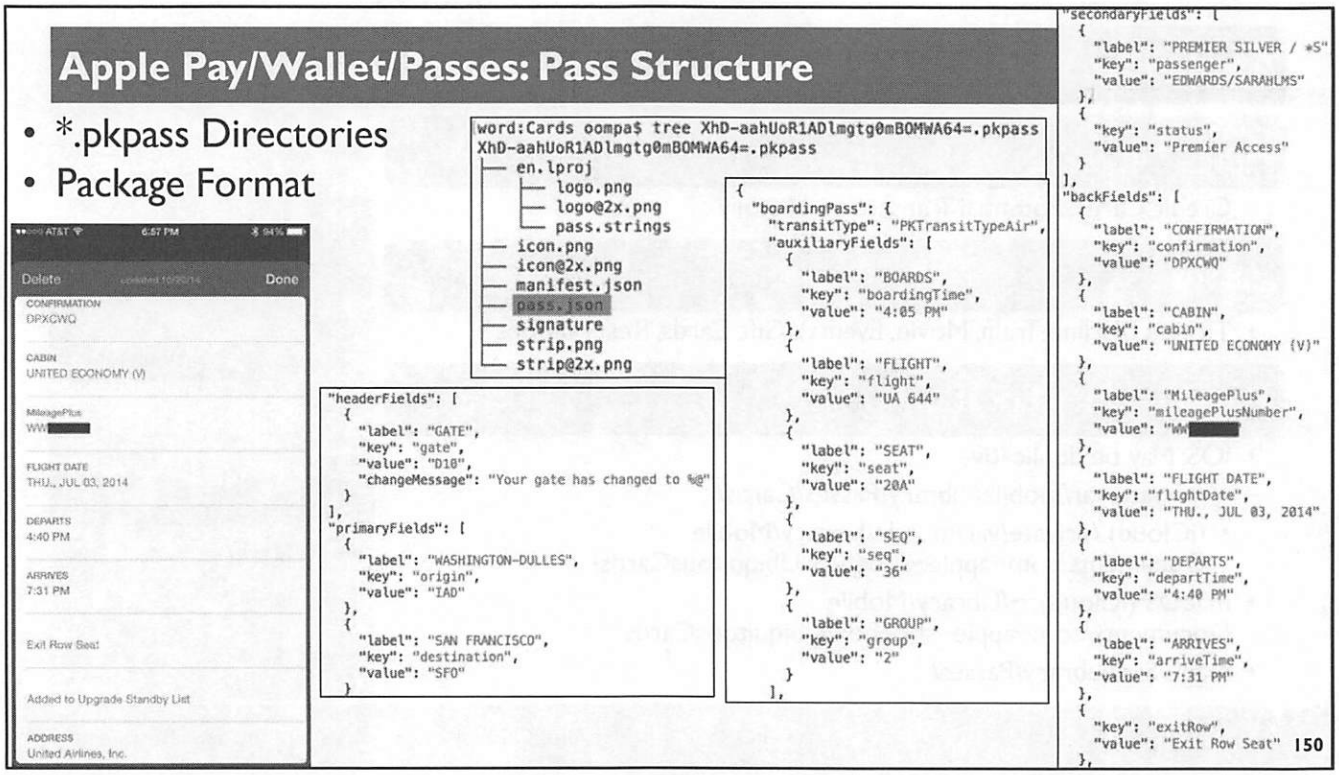
A potential for transaction history may be available if Apple Pay is used in this application. If Wallet/Passes are used, each item is called a "Card". Cards can be anything from an airline boarding pass to a train or event ticket to a booked hotel stay.

These items are synced using iCloud to all devices signed into the user's iCloud account.

For those users with a new Mac with Touch ID, they can now set up credit cards on the Mac itself as shown below.



© 2020 Sarah Edwards

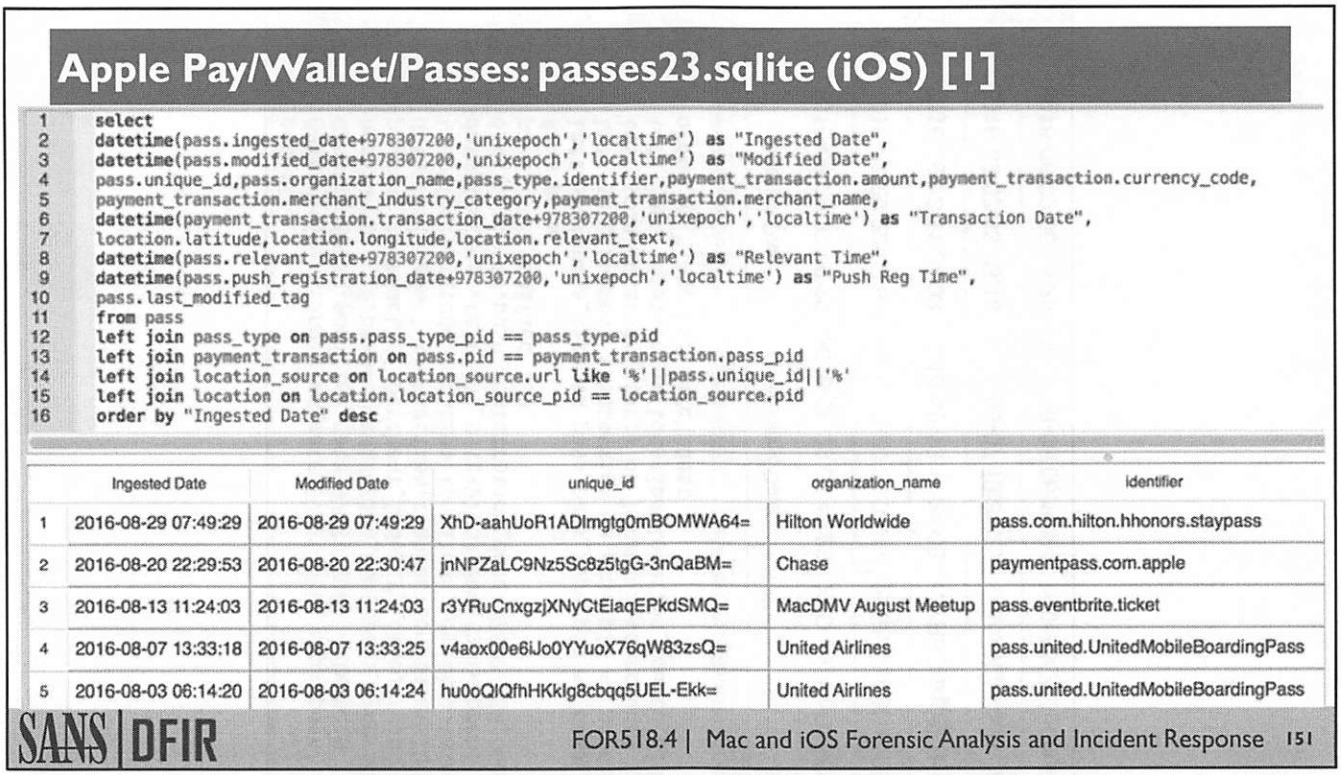


Each "Card" is stored in a `*.pkpass` directory. This directory is actually in a "Package" format. This directory may include lots of files and pictures of various logos associated with that Card. The actual pass data is stored in the pass.json file.

The `pass.json` file can be reviewed in any text editor. Sometimes the developer of the "pass" doesn't include whitespace, so reviewing them can be difficult. Look for an online or offline JSON file formatter.

The example above shows an airline boarding pass's data, which includes to/from locations, seat, timeframes, and other flight information.

© 2020 Sarah Edwards



To extract the information for each Card from the `passes23.sqlite` database, the following SQLite query was used. This query extracts lots of information. This is only a partial view of the extracted data; it is continued on the next slide. The data shown includes the "`ingested data`", when the Card was downloaded to the applications, a last modified timestamp, the unique Card ID, organization name, and bundle identifier. The timestamps are in Mac Epoch format.

```
select
datetime(pass.ingested_date+978307200,'unixepoch','localtime') as "Ingested Date",
datetime(pass.modified_date+978307200,'unixepoch','localtime') as "Modified Date",
pass.unique_id,
pass.organization_name,
pass_type.identifier,
payment_transaction.amount,
payment_transaction.currency_code,
payment_transaction.merchant_industry_category,
payment_transaction.merchant_name,
datetime(payment_transaction.transaction_date+978307200,'unixepoch','localtime') as
"Transaction Date",
location.latitude,
location.longitude,
location.relevant_text,
datetime(pass.relevant_date+978307200,'unixepoch','localtime') as "Relevant Time",
datetime(pass.push_registration_date+978307200,'unixepoch','localtime') as "Push Reg
Time",
pass.last_modified_tag
from pass
left join pass_type on pass.pass_type_pid == pass_type.pid
left join payment_transaction on pass.pid == payment_transaction.pass_pid
left join location_source on location_source.url like '%'||pass.unique_id||'%'
left join location on location.location_source_pid == location_source.pid
order by "Ingested Date" desc
```

© 2020 Sarah Edwards

```
1   select
2   datetime(pass.ingested_date+978307200,'unixepoch','localtime') as "Ingested Date",
3   datetime(pass.modified_date+978307200,'unixepoch','localtime') as "Modified Date",
4   pass.unique_id,pass.organization_name,pass_type.identifier,payment_transaction.amount,payment_transaction.currency_code,
5   payment_transaction.merchant_industry_category,payment_transaction.merchant_name,
6   datetime(payment_transaction.transaction_date+978307200,'unixepoch','localtime') as "Transaction Date",
7   location.latitude,location.longitude,location.relevant_text,
8   datetime(pass.relevant_date+978307200,'unixepoch','localtime') as "Relevant Time",
9   datetime(pass.push_registration_date+978307200,'unixepoch','localtime') as "Push Reg Time",
10  pass.last_modified_tag
11  from pass
12  left join pass_type on pass.pass_type_pid == pass_type.pid
13  left join payment_transaction on pass.pid == payment_transaction.pass_pid
14  left join location_source on location_source.url like '%'||pass.unique_id||'%'
15  left join location on location.location_source_pid == location_source.pid
16  order by "Ingested Date" desc
```

	Ingested Date	Modified Date	unique_id	organization_name	identifier
1	2016-08-29 07:49:29	2016-08-29 07:49:29	XhD-aahUoR1ADImgtg0mBOMWA64=	Hilton Worldwide	pass.com.hilton.hhonors.staypass
2	2016-08-20 22:29:53	2016-08-20 22:30:47	jnNPZaLC9Nz5Sc8z5tgG-3nQaBM=	Chase	paymentpass.com.apple
3	2016-08-13 11:24:03	2016-08-13 11:24:03	r3YRuCnxgzjXNyCtEiaqEPkdSMQ=	MacDMV August Meetup	pass.eventbrite.ticket
4	2016-08-07 13:33:18	2016-08-07 13:33:25	v4aox00e6iJo0YYuoX76qW83zsQ=	United Airlines	pass.united.UnitedMobileBoardingPass
5	2016-08-03 06:14:20	2016-08-03 06:14:24	hu0oQlQfhHKklg8cbqq5UEL-Ekk=	United Airlines	pass.united.UnitedMobileBoardingPass

© 2020 Sarah Edwards

Apple Pay/Wallet/Passes: passes23.sqlite (iOS) [2]

- Apple Pay Transactions and/or Location/Geofence Information

```
select
datetime(pass.ingested_date+978307200,'unixepoch','localtime') as "Ingested Date",
datetime(pass.modified_date+978307200,'unixepoch','localtime') as "Modified Date",
pass.unique_id,pass.organization_name,pass_type.identifier,payment_transaction.amount,payment_transaction.currency_code,
payment_transaction.merchant_industry_category,payment_transaction.merchant_name,
datetime(payment_transaction.transaction_date+978307200,'unixepoch','localtime') as "Transaction Date",
location.latitude,location.longitude,location.relevant_text,
datetime(pass.relevant_date+978307200,'unixepoch','localtime') as "Relevant Time",
datetime(pass.push_registration_date+978307200,'unixepoch','localtime') as "Push Reg Time",
pass.last_modified_tag
from pass
left join pass_type on pass.pass_type_pid == pass_type.pid
left join payment_transaction on pass.pid == payment_transaction.pass_pid
left join location_source on location_source.url like '%'||pass.unique_id||'%'
left join location on location.location_source_pid == location_source.pid
where amount > 0 or latitude > 0
```

	organization_name	identifier	amount	currency_code	merchant_industry_category	merchant_name	Transaction Date	latitude	longitude	relevant_text	Relevant Time
1	Chase	paymentpass.com.apple	2429	USD	SUPERMARKETS	WHOLEFDS ARL 100	2016-08-21 14:41:29	NULL	NULL	NULL	NULL
2	Amtrak	pass.com.amtrak.rider.pnr	NULL	NULL	NULL	NULL	NULL	40.750327	-73.994459	Your departure station is nearby.	2016-07-03 15:00:00
3	Amtrak	pass.com.amtrak.rider.pnr	NULL	NULL	NULL	NULL	NULL	38.896993	-77.006422	Your departure station is nearby.	2016-07-02 06:17:00
4	Lufthansa	pass.com.lufthansa.mbp	NULL	NULL	NULL	NULL	NULL	48.3539	11.7861	Lufthansa LH414 Boarding 15:15	2014-07-26 09:15:00
5	Lufthansa	pass.com.lufthansa.mbp	NULL	NULL	NULL	NULL	NULL	59.6519	17.9186	Lufthansa LH2415 Boarding 11:25	2014-07-26 05:25:00
6	Lufthansa	pass.com.lufthansa.mbp	NULL	NULL	NULL	NULL	NULL	38.9444	-77.4558	Lufthansa LH419 Boarding 17:30	2014-07-17 17:30:00
7	Hilton Worldwide	pass.com.hilton.hhonors.staypass	NULL	NULL	NULL	NULL	NULL	36.858906	-75.97771	Hilton Virginia Beach Oceanfront	NULL

The Apple Pay transactions and location/geofence data are continued here from the previous slide.

The transactions information includes the amount in US dollars (`currency_code`) and shows the vendor information. This data is for a transaction at a Whole Foods supermarket for $24.29.

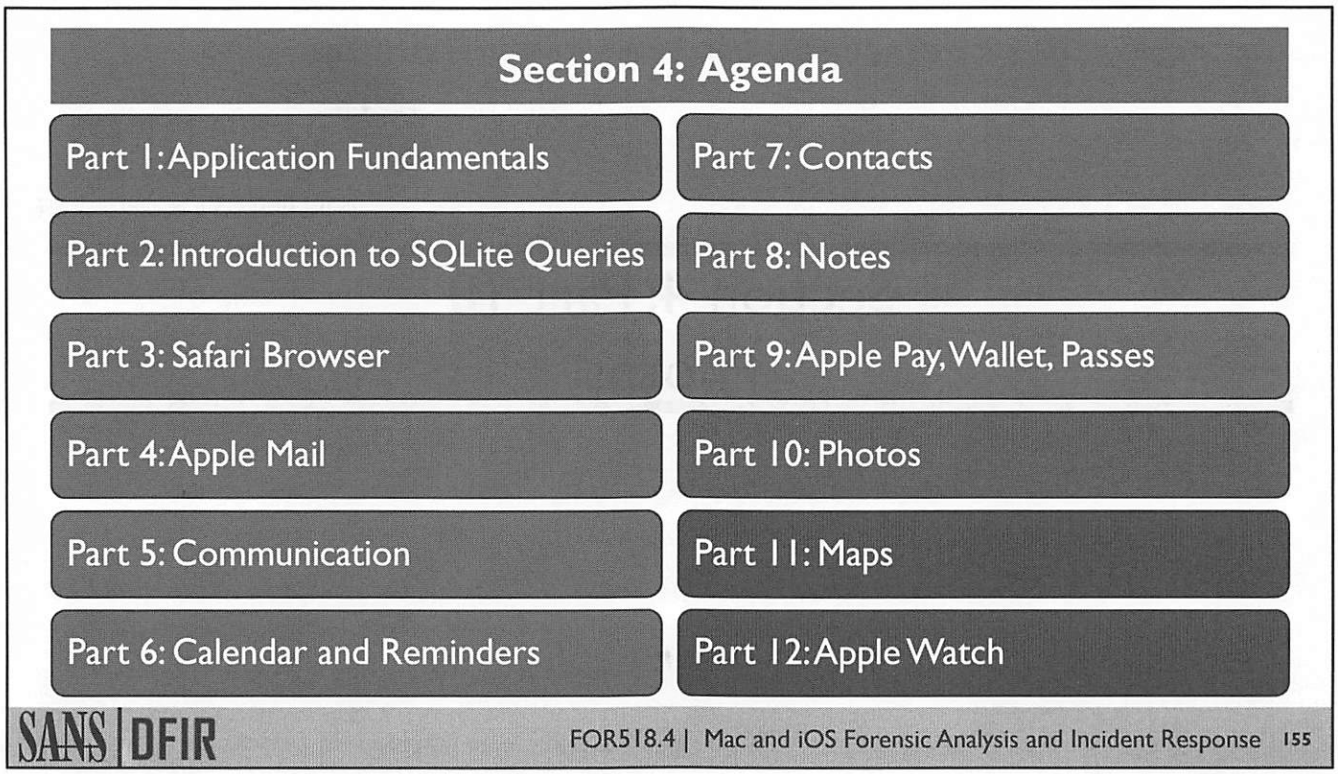
The location and geofence data may appear if the developer of the pass included it so that the pass/ticket would show up on the user's screen when the user entered that specific location at the "`relevant time`".

© 2020 Sarah Edwards

```
select
datetime(pass.ingested_date+978307200,'unixepoch','localtime') as "Ingested Date",
datetime(pass.modified_date+978307200,'unixepoch','localtime') as "Modified Date",
pass.unique_id,pass.organization_name,pass_type.identifier,payment_transaction.amount,payment_transaction.currency_code,
payment_transaction.merchant_industry_category,payment_transaction.merchant_name,
datetime(payment_transaction.transaction_date+978307200,'unixepoch','localtime') as "Transaction Date",
location.latitude,location.longitude,location.relevant_text,
datetime(pass.relevant_date+978307200,'unixepoch','localtime') as "Relevant Time",
datetime(pass.push_registration_date+978307200,'unixepoch','localtime') as "Push Reg Time",
pass.last_modified_tag
from pass
left join pass_type on pass.pass_type_pid == pass_type.pid
left join payment_transaction on pass.pid == payment_transaction.pass_pid
left join location_source on location_source.url like '%'||pass.unique_id||'%'
left join location on location.location_source_pid == location_source.pid
where amount > 0 or latitude > 0
```

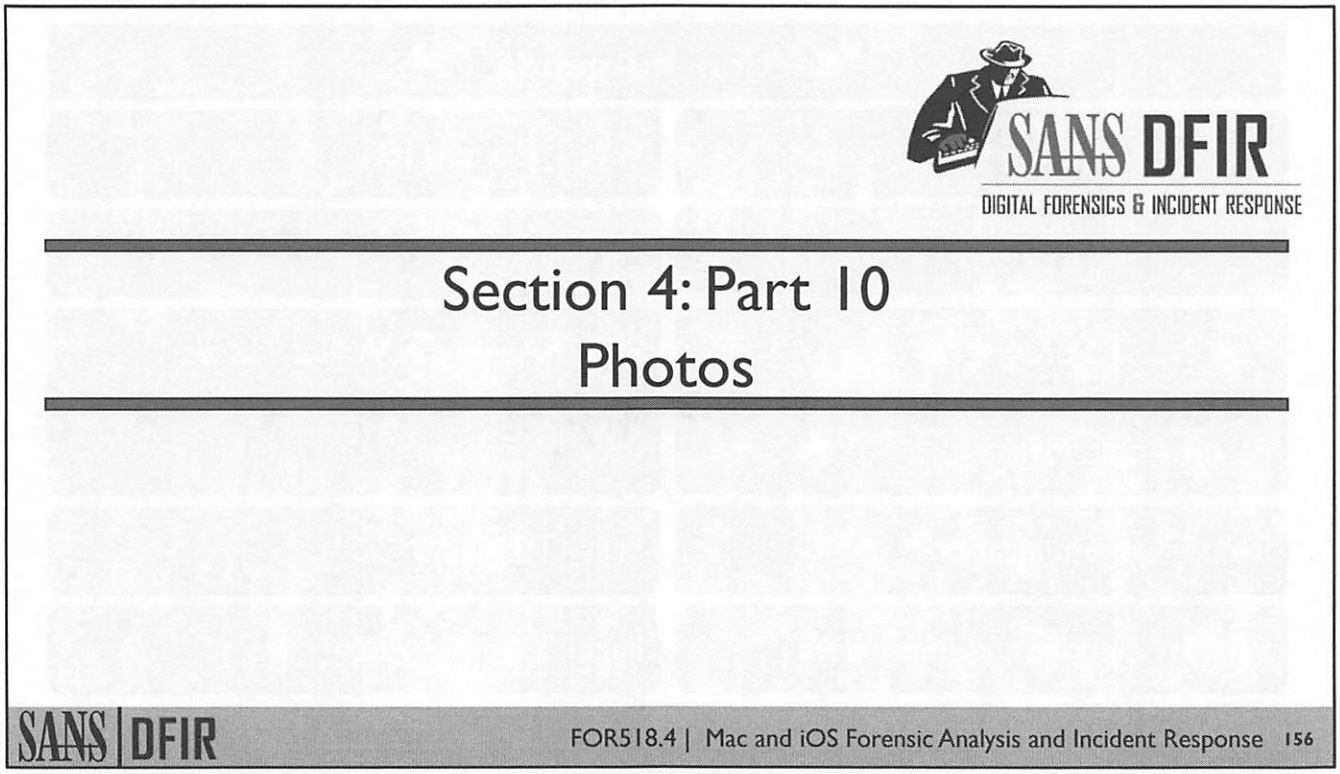
	organization_name	identifier	amount	currency_code	merchant_industry_category	merchant_name	Transaction Date	latitude	longitude	relevant_text	Relevant Time
1	Chase	paymentpass.com.apple	2429	USD	SUPERMARKETS	WHOLEFDS ARL 100	2016-08-21 14:41:29	NULL	NULL	NULL	NULL
2	Amtrak	pass.com.amtrak.rider.pnr	NULL	NULL	NULL	NULL	NULL	40.750327	-73.994459	Your departure station is nearby.	2016-07-03 15:00:00
3	Amtrak	pass.com.amtrak.rider.pnr	NULL	NULL	NULL	NULL	NULL	38.896993	-77.006422	Your departure station is nearby.	2016-07-02 06:17:00
4	Lufthansa	pass.com.lufthansa.mbp	NULL	NULL	NULL	NULL	NULL	48.3539	11.7861	Lufthansa LH414 Boarding 15:15	2014-07-26 09:15:00
5	Lufthansa	pass.com.lufthansa.mbp	NULL	NULL	NULL	NULL	NULL	59.6519	17.9186	Lufthansa LH2415 Boarding 11:25	2014-07-26 05:25:00
6	Lufthansa	pass.com.lufthansa.mbp	NULL	NULL	NULL	NULL	NULL	38.9444	-77.4558	Lufthansa LH419 Boarding 17:30	2014-07-17 17:30:00
7	Hilton Worldwide	pass.com.hilton.hhonors.staypass	NULL	NULL	NULL	NULL	NULL	36.858906	-75.97771	Hilton Virginia Beach Oceanfront	NULL

© 2020 Sarah Edwards



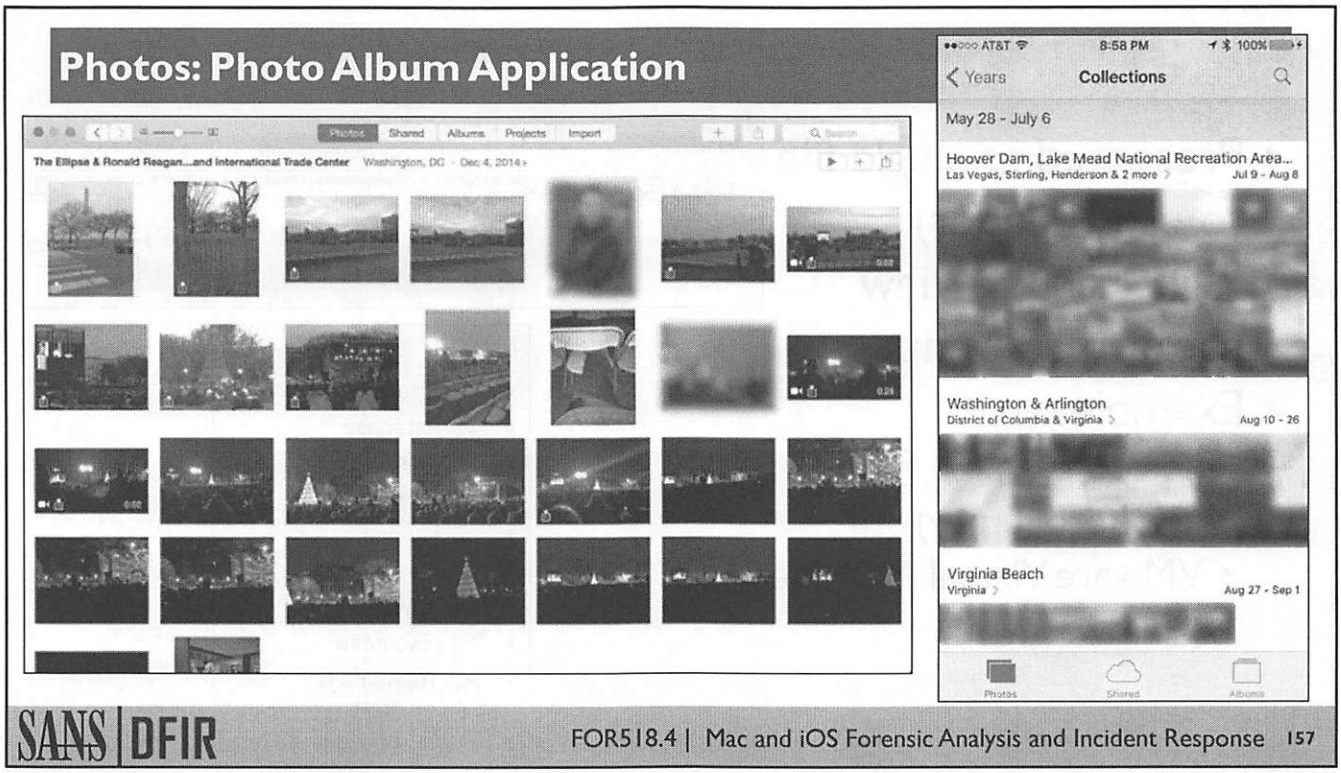
This page intentionally left blank.

© 2020 Sarah Edwards



This page intentionally left blank.

© 2020 Sarah Edwards



The new Photos application changed everything under the hood from its predecessor iPhoto.

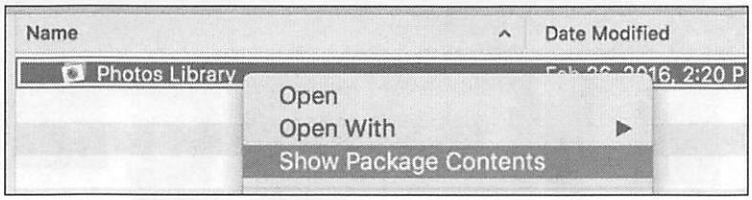
The User's macOS Photo library can be found in a package file: `~/Pictures/Photos Library.photoslibrary/`

Where local photos and iCloud photos were separate before, Photos introduces an integrated format.

© 2020 Sarah Edwards

macOS Photos Library: Packages "File" Format

- Presented as single file to user (Finder)
- Right-click to view package contents
- Examples:
  - iPhoto Library (*.iphotolibrary)
  - VMware Virtual Machines (*.vmwarevm)



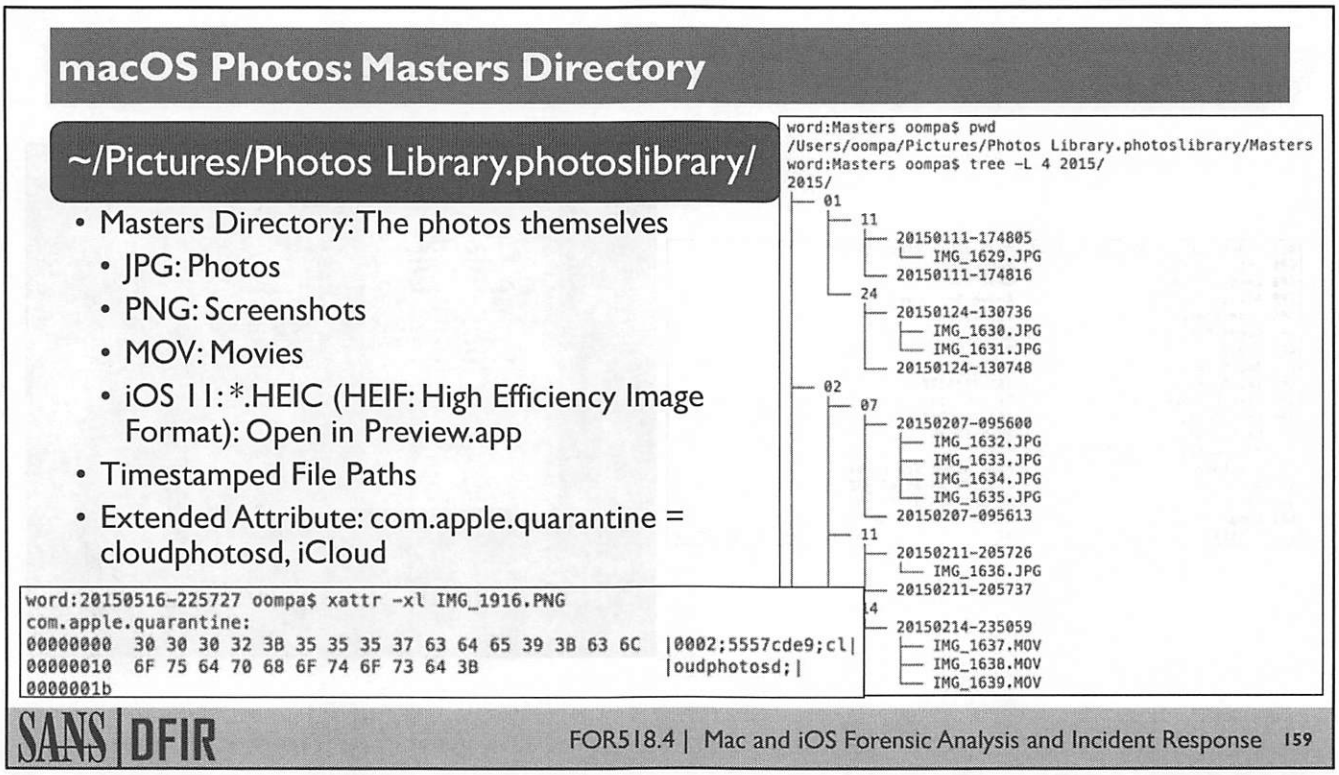
SANS | DFIR FOR518.4 | Mac and iOS Forensic Analysis and Incident Response 158

Packages are directories that are presented to the user as a single file in Finder. To view the contents of a package, right-click and choose "`Show Package Contents`" or view it in a terminal window.

Some of the more well-known packages are the iTunes Library and VMware Fusion virtual machines.

Reference:
Apple Developer Website: About Bundles
`https://developer.apple.com/library/mac/#documentation/CoreFoundation/Conceptual/CFBundles/AboutBundles/AboutBundles.html#//apple_ref/doc/uid/10000123i-CH100-SW1`

© 2020 Sarah Edwards



The original photos can be found in the Masters directory. The photos (and other media) are stored in a timestamped file path.

Since local and iCloud photos are now stored together, it might be necessary to determine which are iCloud pictures. This can be shown in the extended attributes for each file. If the `com.apple.quarantine` attribute contains `cloudphotosd` as its Bundle ID, then it came from iCloud.

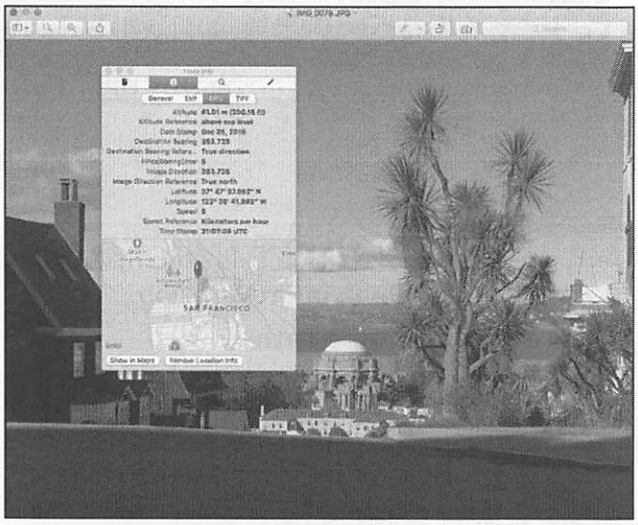
Pictures taken with devices running iOS 11 will use a new format for storage called High Efficiency Image Format.

Reference:
`https://en.wikipedia.org/wiki/High_Efficiency_Image_File_Format`

© 2020 Sarah Edwards

Photo EXIF Data

```
byte:100APPLE oompa$ exiftool IMG_0079.JPG | grep GPS
GPS Latitude Ref                : North
GPS Longitude Ref               : West
GPS Altitude Ref                : Above Sea Level
GPS Time Stamp                  : 21:07:06
GPS Speed Ref                   : km/h
GPS Speed                       : 0
GPS Img Direction Ref           : True North
GPS Img Direction               : 353.7246377
GPS Dest Bearing Ref            : True North
GPS Dest Bearing                : 353.7246377
GPS Date Stamp                  : 2015:12:25
GPS Horizontal Positioning Error: 5 m
GPS Altitude                    : 61 m Above Sea Level
GPS Date/Time                   : 2015:12:25 21:07:06Z
GPS Latitude                    : 37 deg 47' 37.96" N
GPS Longitude                   : 122 deg 26' 41.89" W
GPS Position                    : 37 deg 47' 37.96" N, 122 deg 26' 41.89" W
```



Photos found on iOS devices may contain GPS location coordinates if the user has authorized the application or Camera app to do so.

Two ways to show this information are shown; however, many applications and forensic software suites can provide similar output. On the left is an example of the output of "`exiftool`", a command-line utility for parsing EXIF data, while on the right is a screenshot using the Preview applications Data Inspector to view the same data.

© 2020 Sarah Edwards

Photo Databases

macOS

- Older (10.10): ~/Pictures/Photos Library.photoslibrary/Databases/Library.apdb (Link to /apdb/Library.apdb)
- Newer (10.11+): ~/Pictures/Photos Library.photoslibrary/database/photos.db

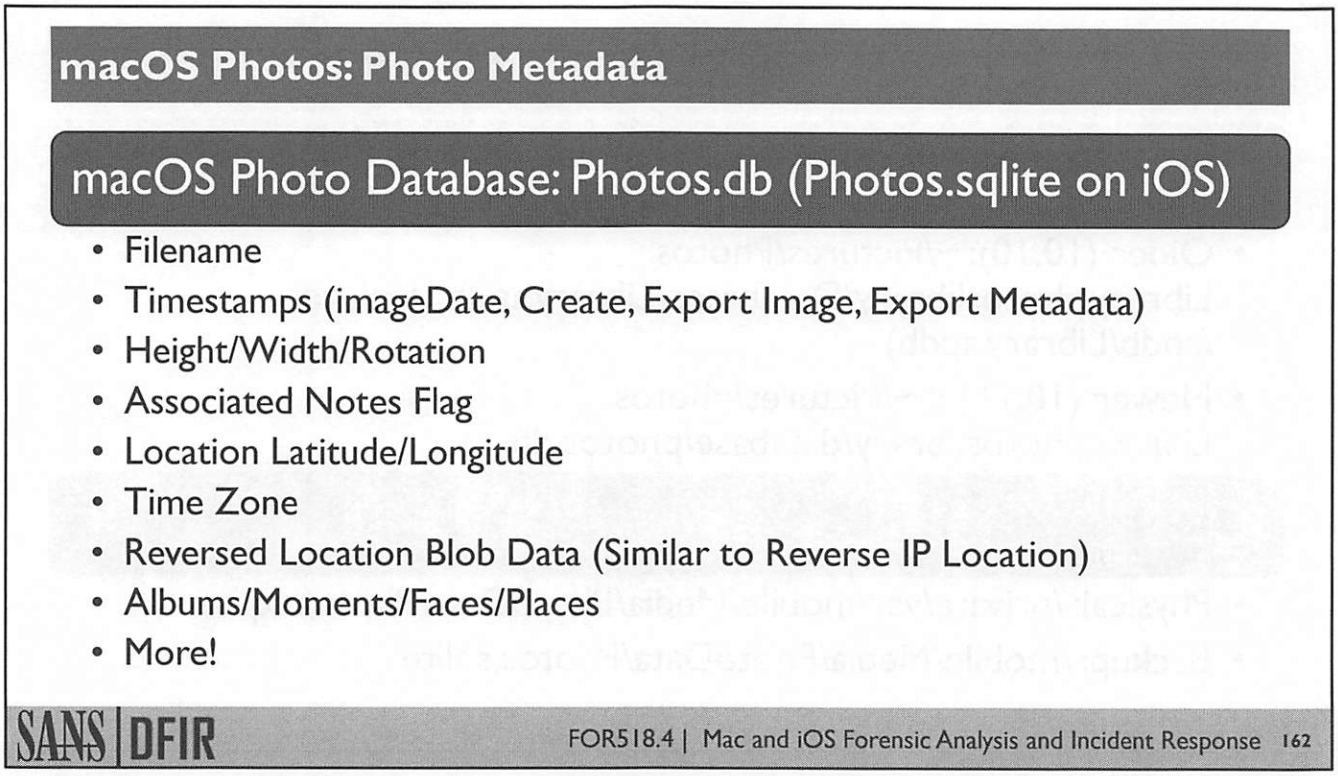
iOS:

- Physical: /private/var/mobile/Media/PhotoData/Photos.sqlite
- Backup: /mobile/Media/PhotoData/Photos.sqlite

The metadata for each photo (or other media) is found in the `Library.apdb/Photos.db/Photos.sqlite` databases. These databases contain nearly the same type of data; however, the table and column names may differ.

There are many other files in the Photos.photoslibrary package file on macOS and the PhotoData directory on iOS, including other databases, plist files, thumbnail photos, etc.

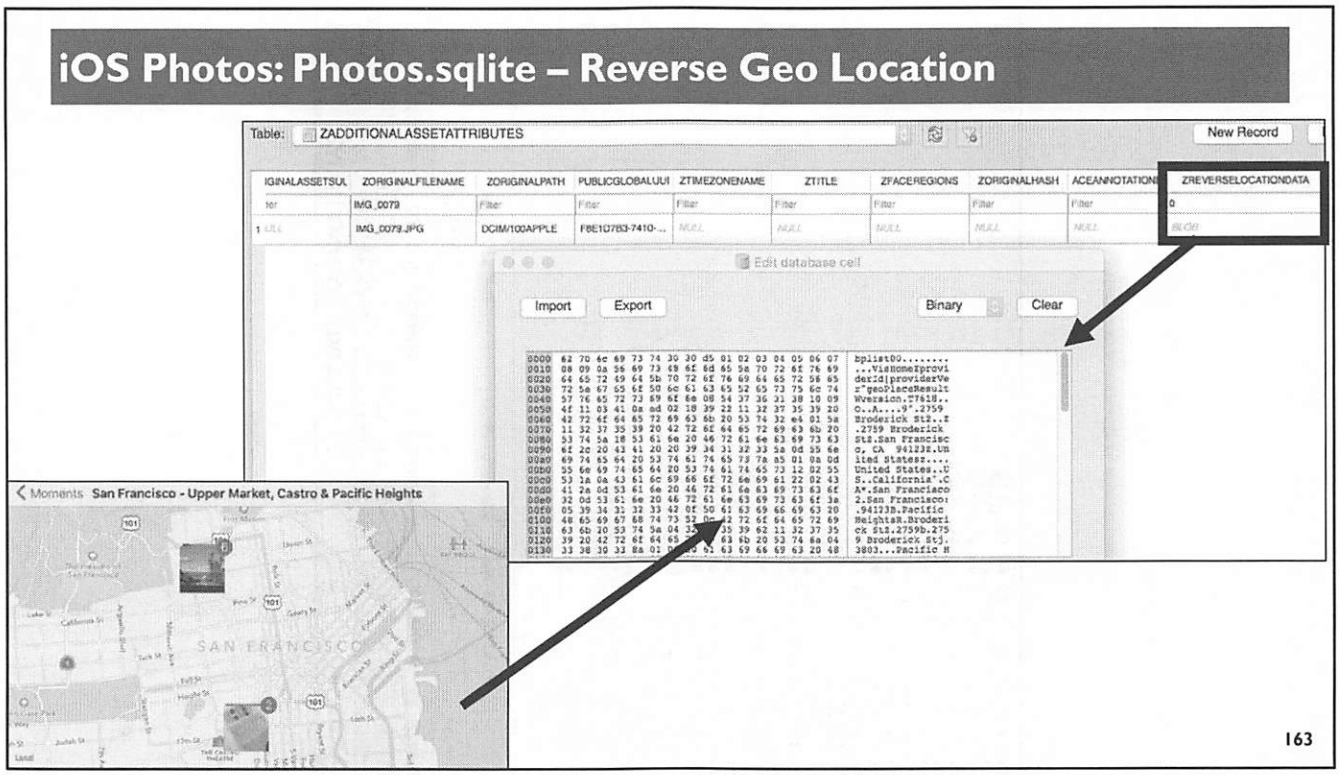
© 2020 Sarah Edwards



macOS: `~/Pictures/Photos Library.photoslibrary/database/photos.db`
iOS Physical: `/private/var/mobile/Media/PhotoData/Photos.sqlite`
iOS Backup: `/mobile/Media/PhotoData/Photos.sqlite`

An example from the `Library.apdb` database contains the metadata for each photo item. This can include a variety of different items such as file metadata, timestamps, time zone info, user comments, and location-based data, just to include some of what is available.

© 2020 Sarah Edwards

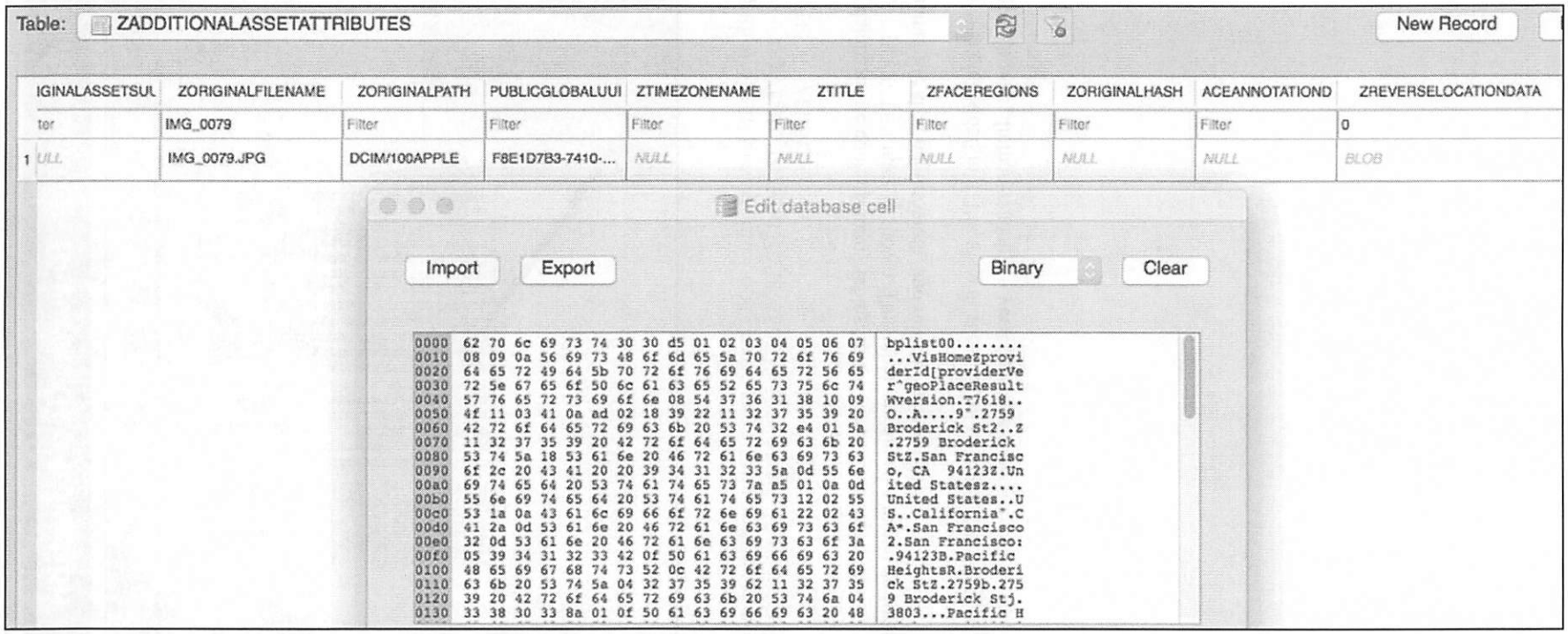


macOS: `~/Pictures/Photos Library.photoslibrary/database/photos.db`
iOS Physical: `/private/var/mobile/Media/PhotoData/Photos.sqlite`
iOS Backup: `/mobile/Media/PhotoData/Photos.sqlite`

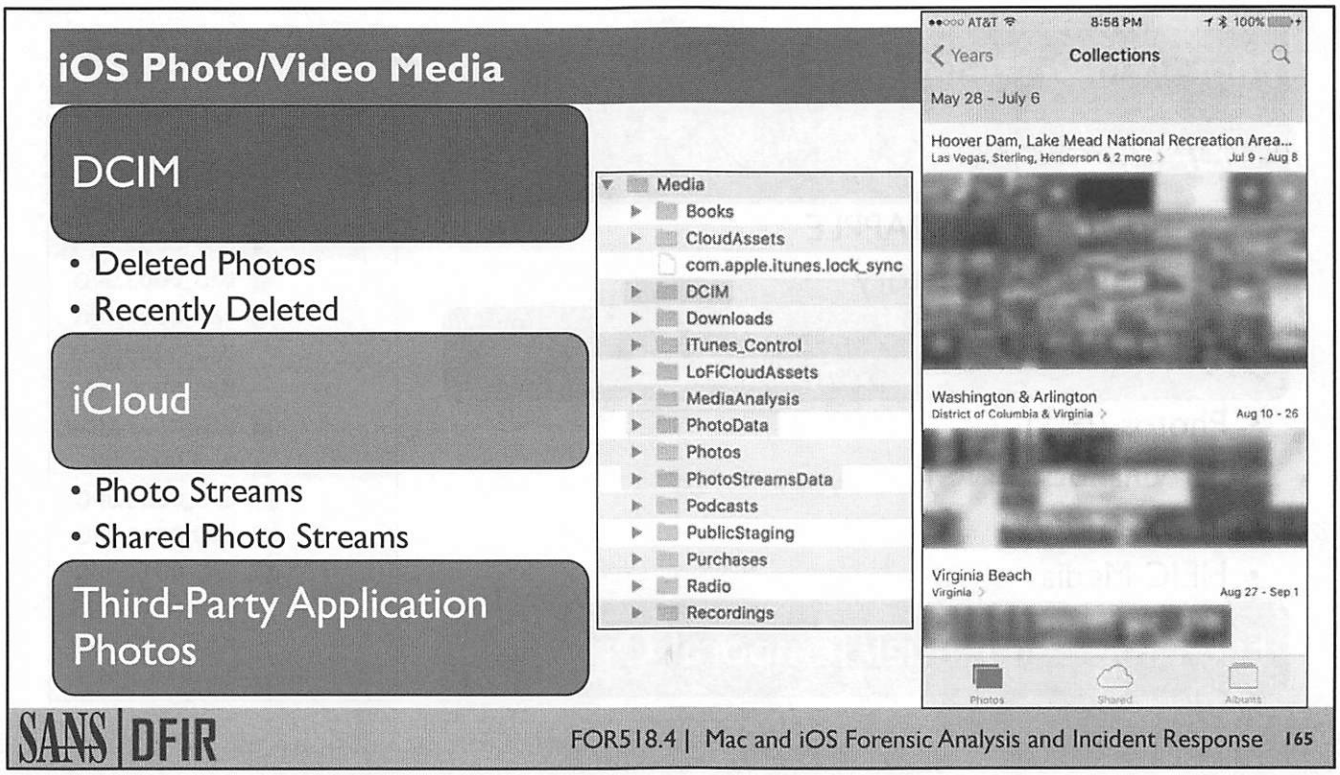
The `Photos.sqlite` database file contains many tables. In the screenshot above, the `ZADDITIONALASSETATTRIBUTES` table is shown. This table contains the metadata for each photo in the database to include filename, file size, height/width, and reverse location data.

If the photo has location data embedded in it, Photos will show a BLOB in the `ZREVERSELOCATIONDATA` table. This BLOB contains a plist showing human-readable reversed geolocation data.

© 2020 Sarah Edwards



© 2020 Sarah Edwards

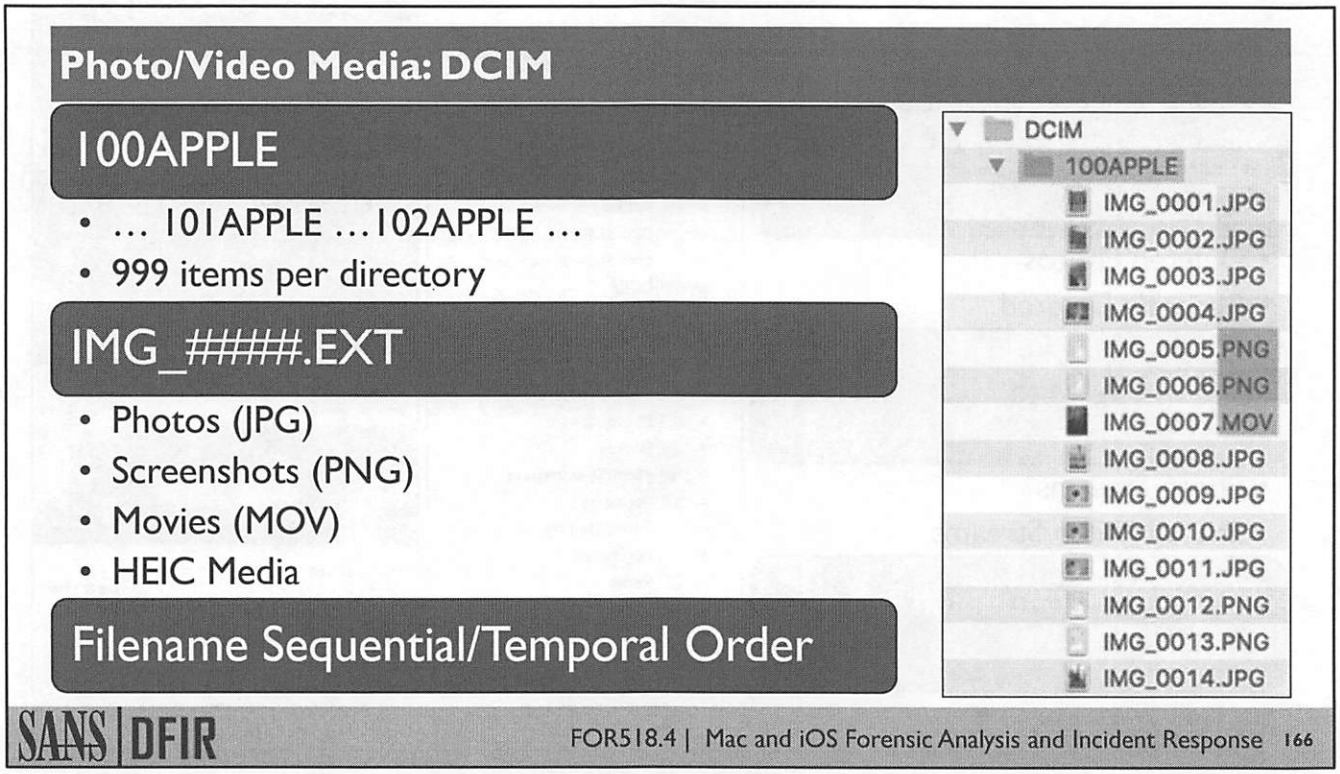


Photos, videos, and screenshots can be strewn about a device in many directories, depending on how the user uses their device.

These directories include the DCIM, PhotoData, and PhotoStreamsData directories.

The native Photos application is shown to the right.

© 2020 Sarah Edwards

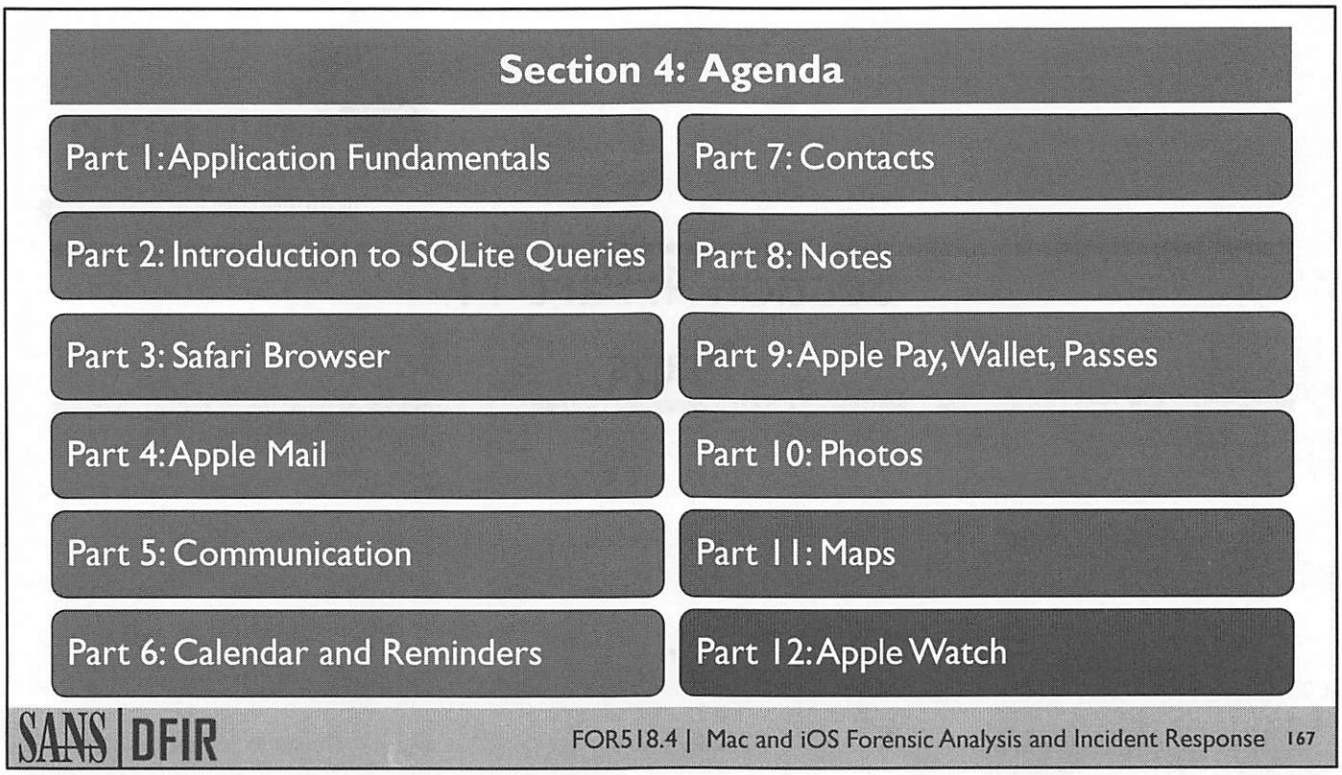


Physical: iOS 6+: `/private/var/mobile/Media/DCIM/`
Backup/FS: iOS 6+: `/mobile/Media/DCIM/`

Photos, videos, and screenshots that are taken with the device will be stored in the DCIM directory. This directory works just like it does on other camera devices. This directory may contain multiple other directories named `100APPLE`, `101APPLE`, and so on. Each of these may contain up to 999 photos, videos, and/or screenshots.

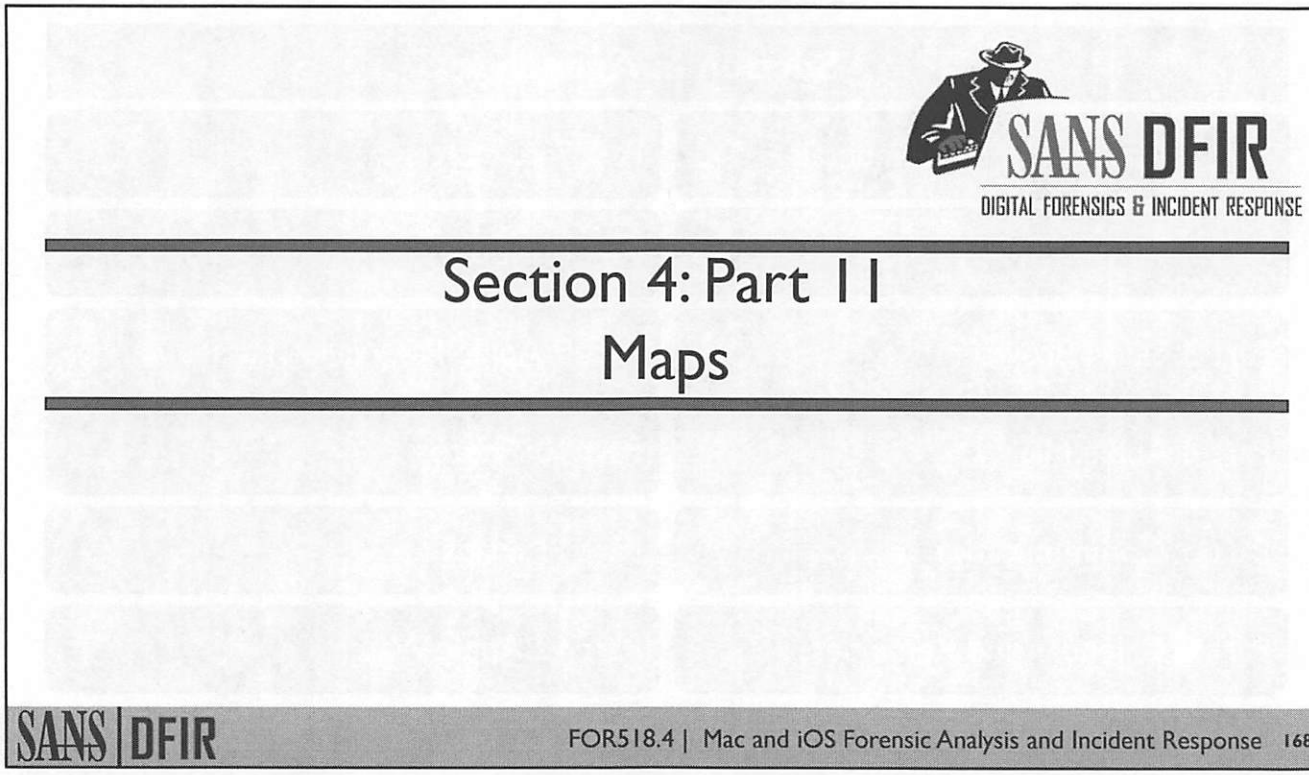
Each image is named sequentially starting with `IMG_0001.EXT`, where EXT can be JPG for a photo, MOV for a movie, or PNG for a screenshot.

© 2020 Sarah Edwards



This page intentionally left blank.

© 2020 Sarah Edwards



This page intentionally left blank.

© 2020 Sarah Edwards

Maps: Files and Directories

iOS Physical

- [iOS 8+] /private/var/mobile/Containers/Data/<GUID>/Library/Maps/

iOS Backup/File System

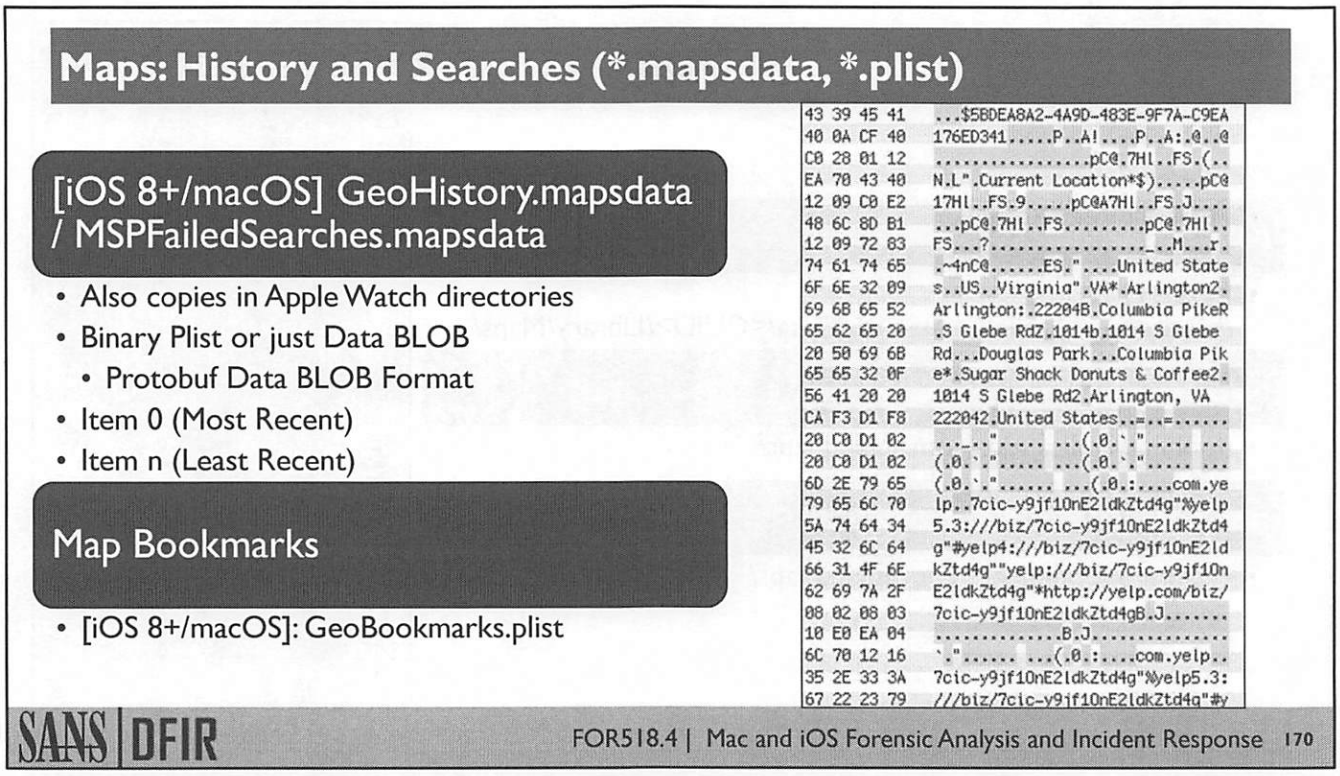
- /mobile/Applications/com.apple.Maps/

macOS

- ~/Library/Containers/com.apple.Maps/

The native mapping application is Apple Maps. The Maps application is available for both iOS and macOS, and the files are very similar. Maps data can be synced to all devices using iCloud.

© 2020 Sarah Edwards



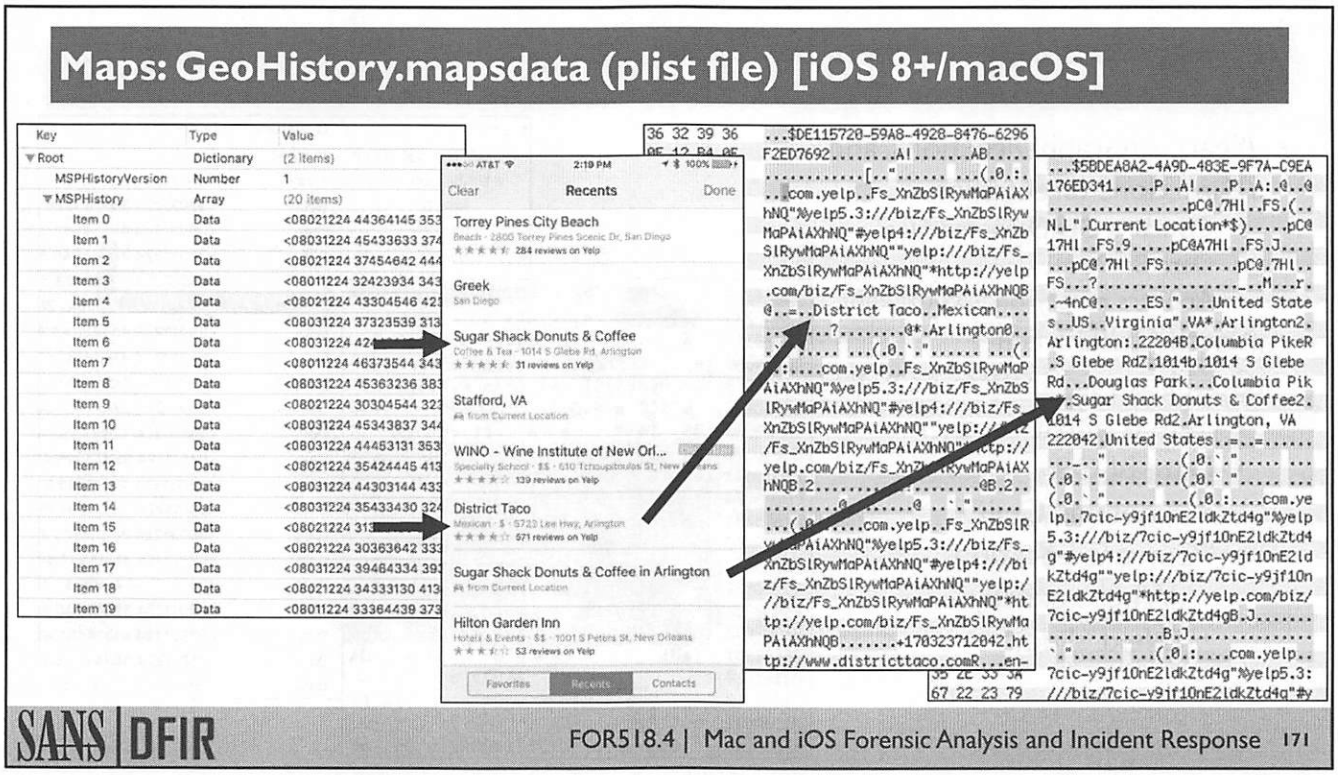
iOS 8+/macOS: `GeoHistory.mapsdata, MSPFailedSearches.mapsdata`
iOS 7-: `History.mapsdata/Directions.mapsdata`

In iOS 7 and older devices, the maps data is stored in a proprietary format file. These files may still exist when a user upgrades to newer operating systems. The ASCII location strings can be viewed either in a hex editor or using the "`strings`" command. The format of these files is not documented.

On newer iOS 8+ devices, the `*.mapsdata` files are stored in a slightly easier-to-read plist file but still contain BLOB location data that is hard to parse. The BLOB data can be extracted and viewed in a hex editor to determine the location data.

It is worth noting that non-ASCII Unicode strings used for other languages may be harder to read using a hex editor or the "`strings`" command.

© 2020 Sarah Edwards



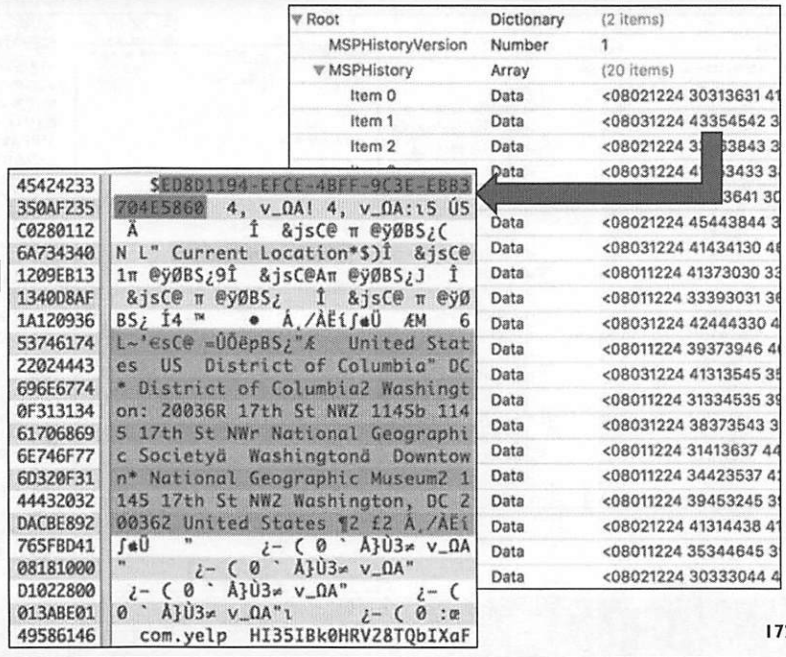
The `GeoHistory.mapsdata` plist file contains individual history items but in the proprietary BLOB format. Each item may contain a start and finish route, possible Yelp information, and/or reverse geolocation data.

Each item contains a GUID in the first few bytes of the BLOB that can be used to potentially correlate to an on-disk file with a matching GUID, as shown in future slides.

© 2020 Sarah Edwards

Maps: Searches and Directions [1]

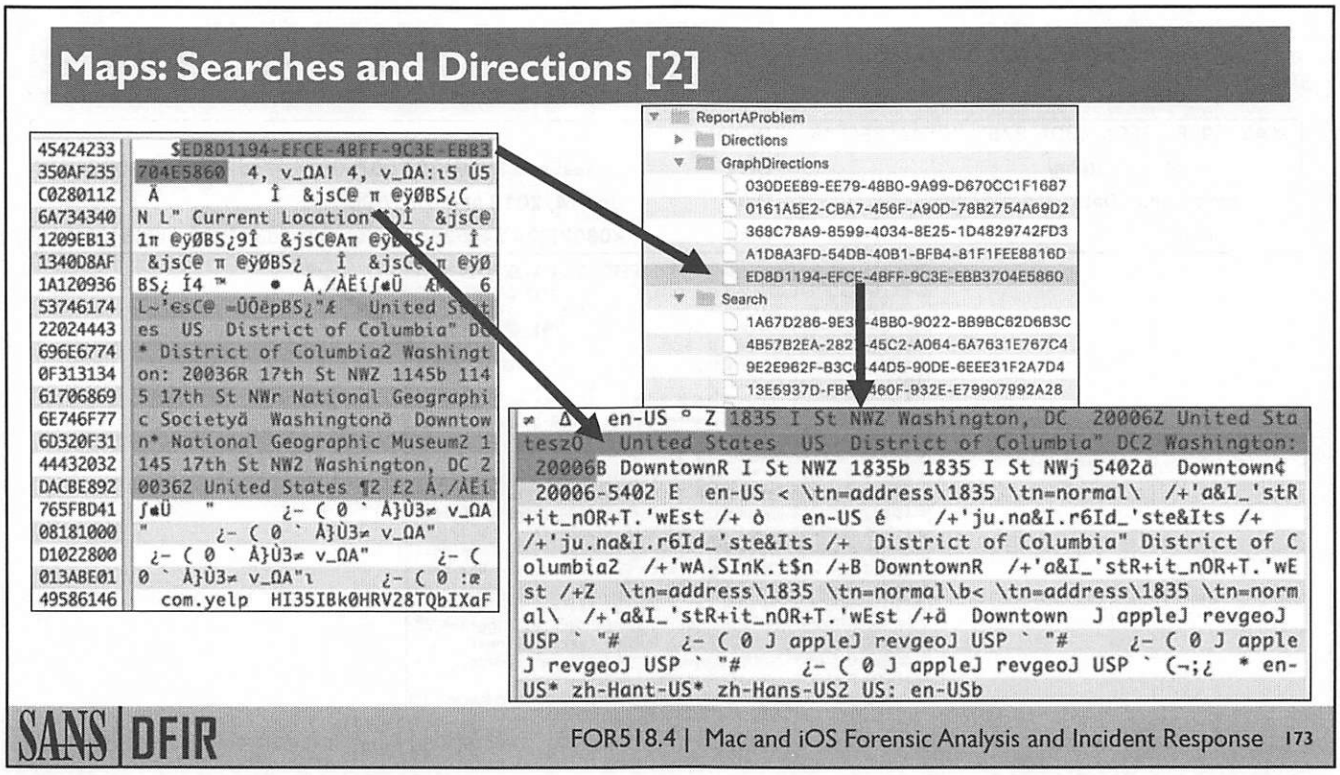
- /Search, /GraphDirections, and /Directions directories
  - Contains GUID-named files
  - Data BLOBs
  - Appears empty on iOS backups
- Match GUID from GeoHistory.mapsdata [iOS 8+]
  - Item 0 (most recent)
  - Data BLOB
    - Contains GUID
  - "Current Location"
    - Where exactly?
    - (Next slide)



The GUID in each `GeoHistory.mapsdata` item can be matched to files in the `/Search`, `/GraphDirections`, or `/Direction` directories to find additional related information.

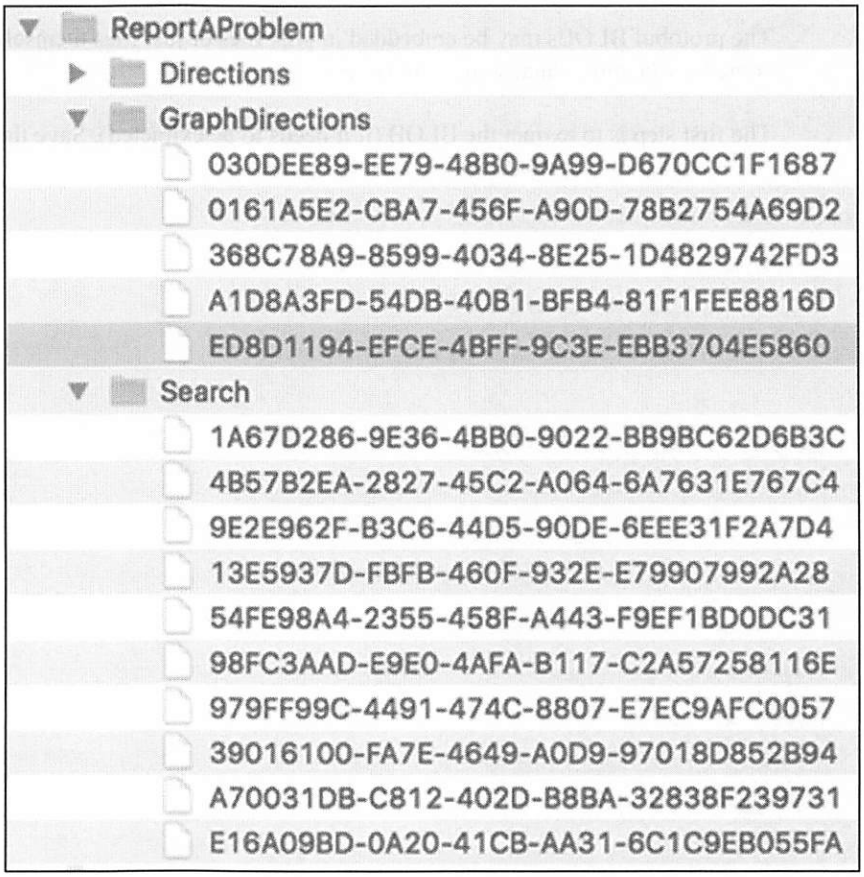
These files do not appear to get backed up in iTunes backups.

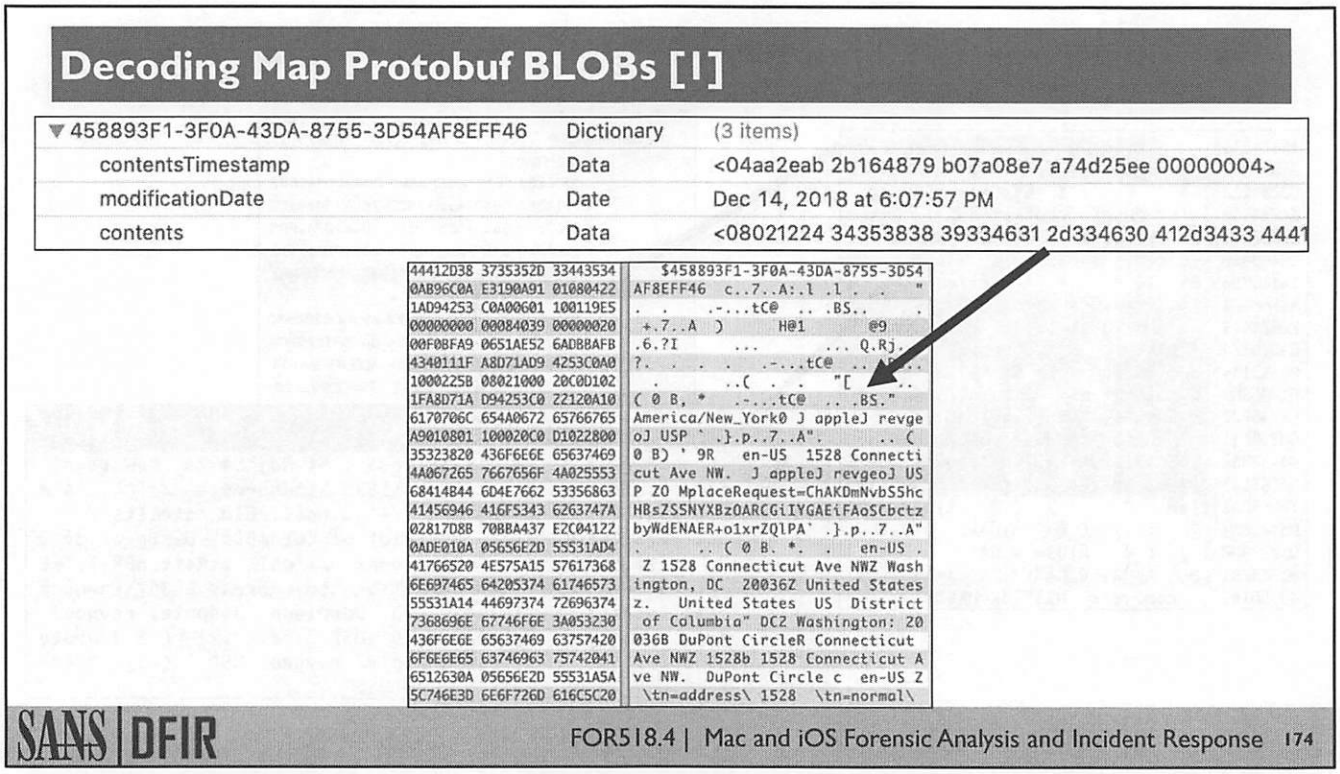
© 2020 Sarah Edwards



These GUID files can contain route data in the `/Directions` and `/GraphDirections` directories and possibly Yelp and other search data in the `/Search` directory.

In the example above, this shows a map route from "`Current Location`" to "`National Geographic Museum`" in Washington, DC. Toward the end of this file, there will be reverse geolocation data (look for "`revgeo`" strings) that may be able to show you the start location of "`Current Location`". In the example, this is "`1835 I Street`" in "`Washington, DC`".





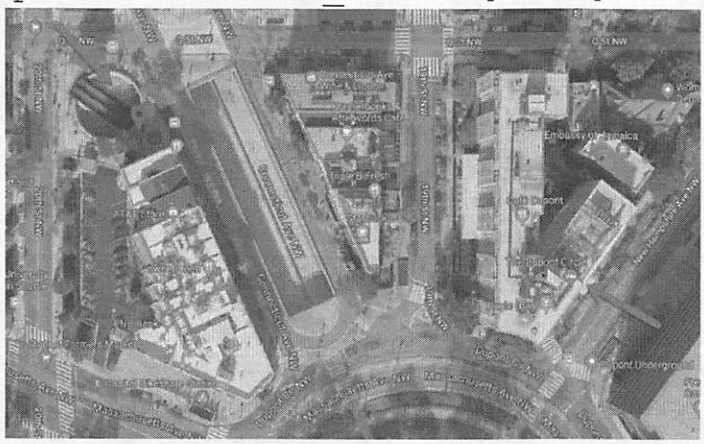
The protobuf BLOBs may be embedded in plist files or just files themselves. They look like the example above; some may be more data, some may be less.

The first step is to extract the BLOB (if it needs to be extracted). Save this data to a file.

© 2020 Sarah Edwards

Decoding Map Protobuf BLOBs [2]

- Use 'protoc' utility to decode
  - `brew install protobuf`
  - `protoc -decode_raw < [BLOB]`



```
Sarahs-Air:Edits oompa$ protoc --decode_raw < mapblob.txt
1: 2
2: "468893F1-3F0A-43DA-8755-3D54AF8EFF46"
3: 0x41c0e237a4cd6313
7 {
  1 {
    1 {
      1 {
        1: 4
        4 {
          1 {
            1 {
              1: 0x40437496bccd2db7
              2: 0xc05342d91ad7a81f
              100: 1
            }
            2: 1
            3: 0x41c0e237a42b0ae5
            4: 31
            5: 0x4048000000000001
            6: 0x4000000000000000
            7: 0x3fa036ba20000000
            9: 0xbff0000000000000
            100: 0xbff0000000000000
            101: 0x3ffbbadb6a52ae51
            102: 0
            103: 0
            105: 1
            106 {
              1: 0x40437496bccd2db7
              2: 0xc05342d91ad7a81f
              100: 1
            }
            107: 0xbff0000000000000
          }
        }
        5: 1
      }
      2 {
        1 {
          2: 0
          4 {
```

Starting Lat/Long

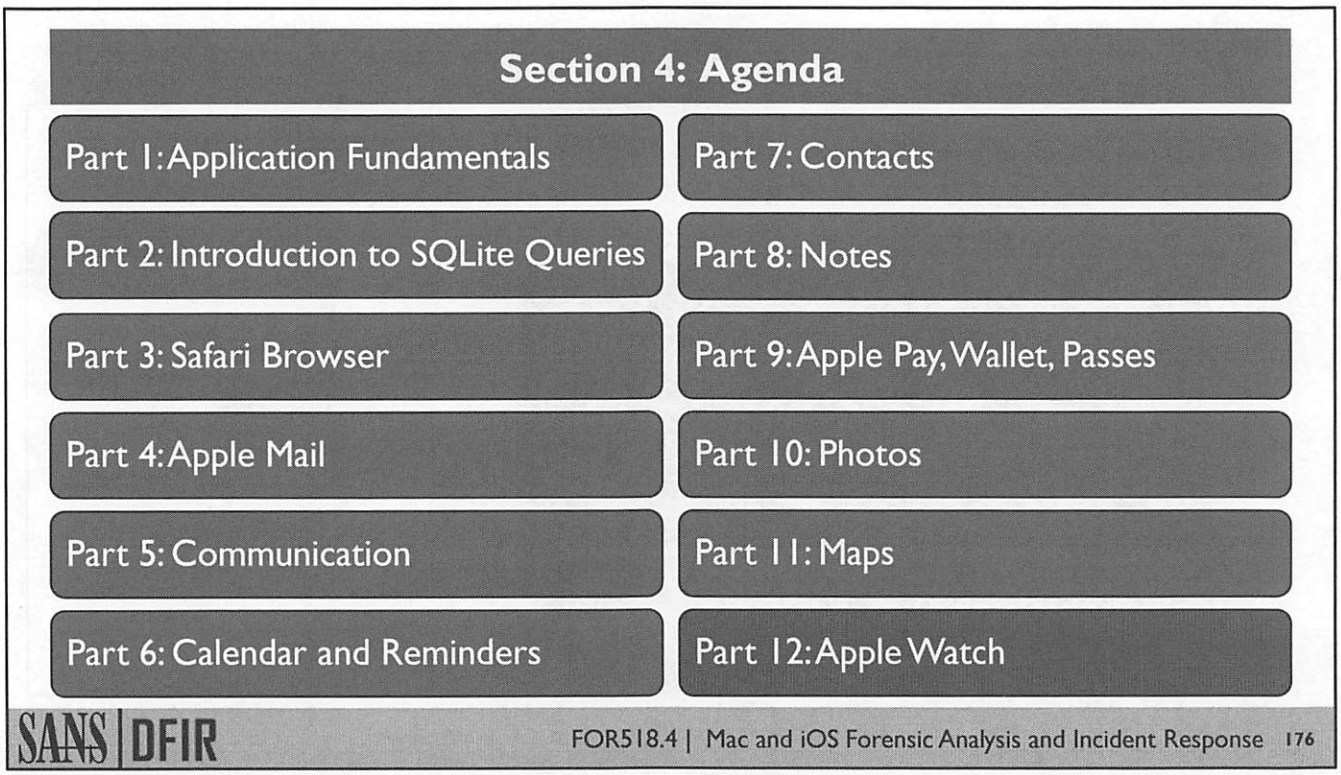
Timestamp

Protobuf data can be decoded by using the protoc utility from Google's Protobuf utilities.

Shown on the right is a partial example of the JSON-like output. There are many pieces of data stored, including originating coordinates (Dupont Metro in DC) and the location search timestamp. The output may be quite verbose; it is advisable to save this output to another file for analysis.

The coordinates were extracted and changed to decimal format and mapped. The Pin on the left shows where the search was originally started from to find directions to the Dupont Circle Hotel.

© 2020 Sarah Edwards



This page intentionally left blank.

© 2020 Sarah Edwards

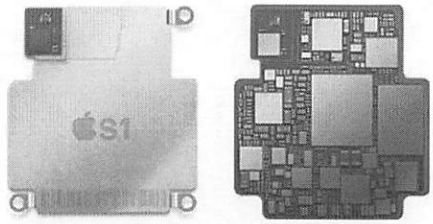
Section 4: Part 12
Apple Watch

This page intentionally left blank.

© 2020 Sarah Edwards

Apple Watch

- Apple's wearable technology
  - "Smart Watch"
- Must be paired with iPhone (iOS 8.2, iPhone 5, or newer)
- Wi-Fi, GPS, NFC, Bluetooth, cellular
- "System in Package"
- Proxy device

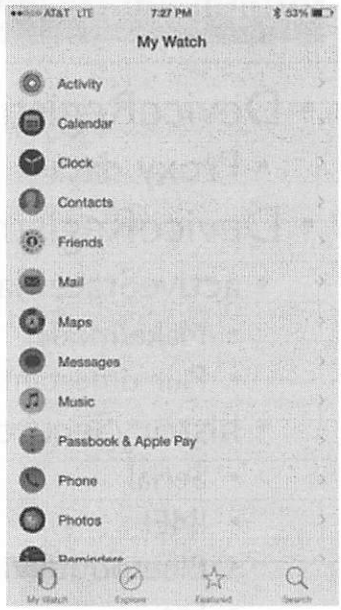


The Apple Watch is Apple's new wearable technology, or "Smart Watch". It comes in two sizes and must be paired with an iPhone (not an iPod or iPad).

© 2020 Sarah Edwards

Software

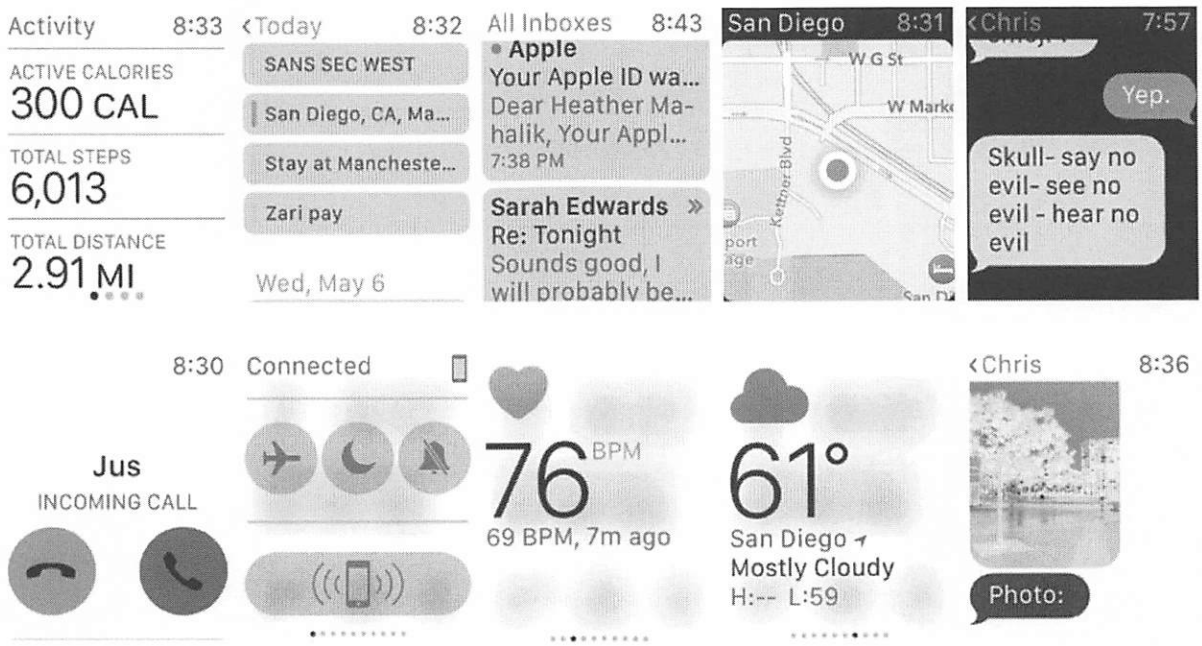
- Software
  - WatchOS: iOS-Based
  - Carousel: The Interface
    - i.e., Springboard/Finder
- Apps:
  - Native: Activity, Calendar, Clock, Contacts, Friends, Mail, Maps, Messages, Music, Passbook/Apple Pay, Phone, Photos, Reminders, Stocks, Weather, Workout
  - Third-party Apps!
- Use the Apple Watch App on iPhone to sync data



179

The Apple Watch runs WatchOS. Research shows this may be a slightly modified version of iOS.

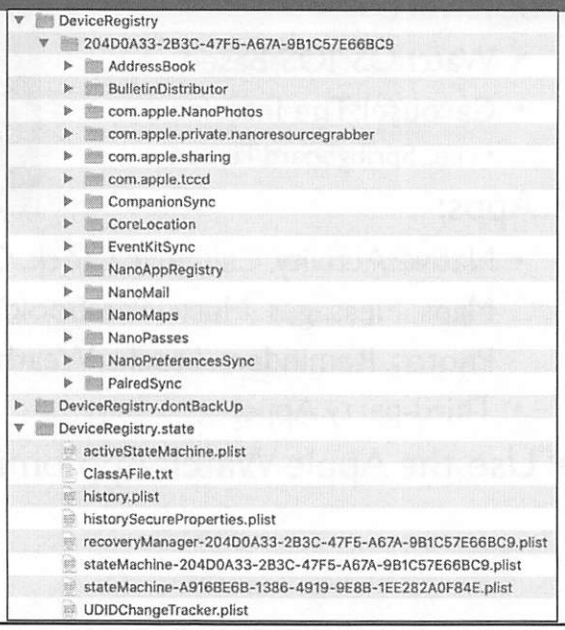
Applications on the watch include many of those on the iPhone. The Watch can be thought of as an extension of the iPhone.



© 2020 Sarah Edwards

Apple Watch: Data Directory /mobile/Library/DeviceRegistry/<GUID>

- DeviceRegistry
  - Proxy data
- DeviceRegistry.state/
  - activeStateMachine.plist
    - Make/model
    - Pair timestamp
  - historySecureProperties.plist
    - Serial
    - IMEI
    - Bluetooth/Wi-Fi MAC addresses
    - Etc.

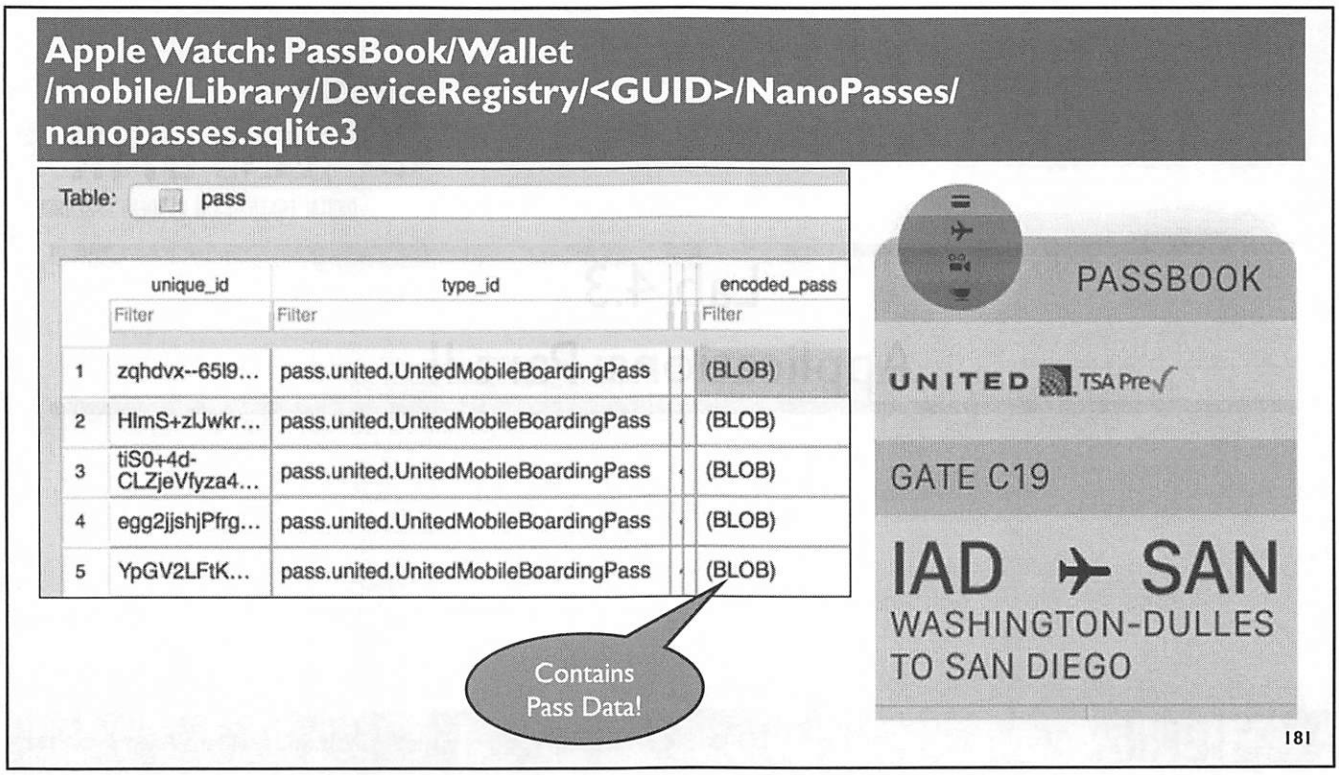


180

Most of the data associated with the Apple Watch in a backup is stored in the `/mobile/Library/DeviceRegistry/` directory. This data is mostly redundant data found on the device itself.

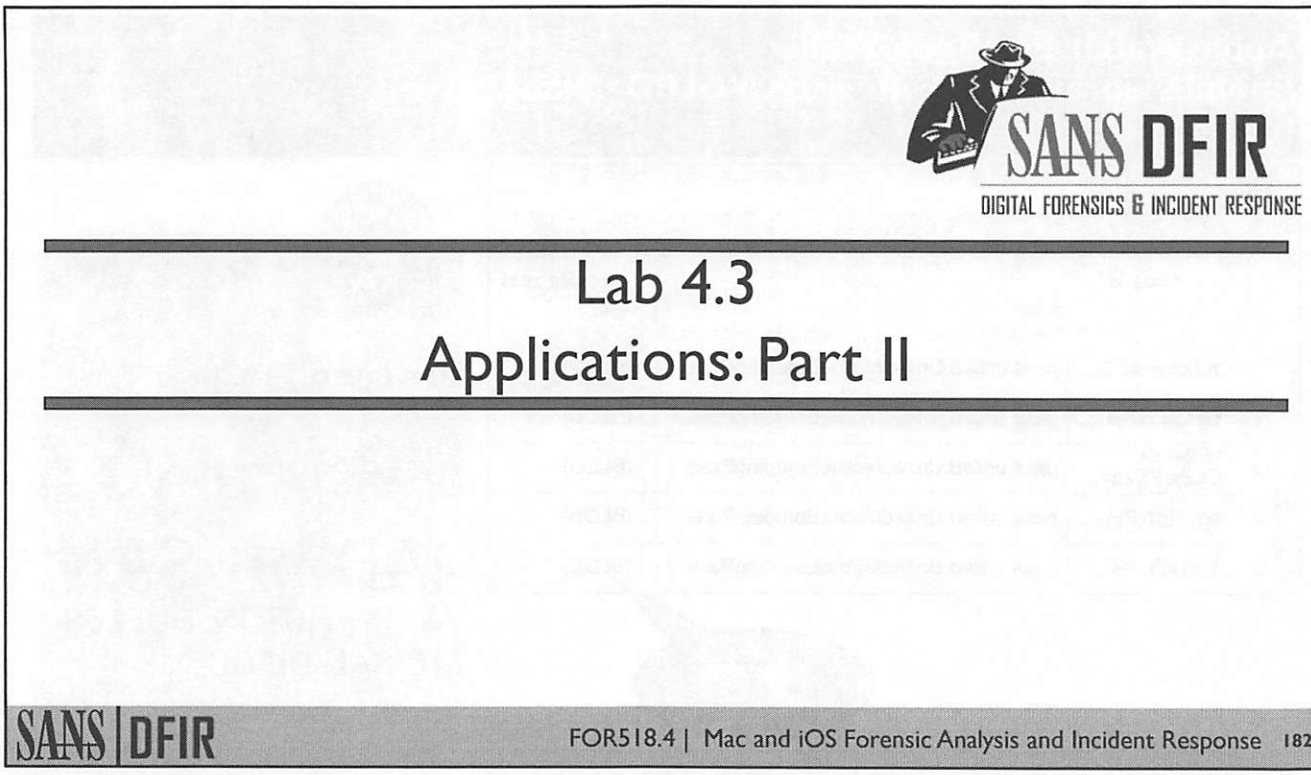
Identifying information can be found in the `DeviceRegistry.state/` directory in a few plist files.

© 2020 Sarah Edwards



The Apple Watch can be used to board flights using the Passbook application. This information is stored in its own SQLite database, `nanopasses.sqlite3`.

© 2020 Sarah Edwards



This page intentionally left blank.

© 2020 Sarah Edwards

FOR518.4 | Mac and iOS Forensic Analysis and Incident Response

FOR518 Section 4: Application Data Analysis

© 2020 Sarah Edwards | All Rights Reserved | Version F01_01

Author: Sarah Edwards
oompa@csh.rit.edu
mac4n6.com
http://twitter.com/iamevltwin

https://digital-forensics.sans.org/
http://twitter.com/sansforensics

"As usual, SANS courses pay for themselves by Day 2. By Day 3, you are itching to get back to the office to use what you've learned."
Ken Evans, Hewlett Packard Enterprise - Digital Investigation Services

SANS Programs
sans.org/programs

GIAC Certifications
Graduate Degree Programs
NetWars & CyberCity Ranges
Cyber Guardian
Security Awareness Training
CyberTalent Management
Group/Enterprise Purchase Arrangements
DoDD 8140
Community of Interest for NetSec
Cybersecurity Innovation Awards

Search SANSInstitute

SANS Free Resources
sans.org/security-resources

- E-Newsletters
  NewsBites: Bi-weekly digest of top news
  OUCH!: Monthly security awareness newsletter
  @RISK: Weekly summary of threats & mitigations
- Internet Storm Center
- CIS Critical Security Controls
- Blogs
- Security Posters
- Webcasts
- InfoSec Reading Room
- Top 25 Software Errors
- Security Policies
- Intrusion Detection FAQ
- Tip of the Day
- 20 Coolest Careers
- Security Glossary

SANS Institute
8120 Woodmont Avenue | Suite 310
Bethesda, MD 20814
301.654.SANS(7267)
info@sans.org