FORENSICATE DIFFERENTLY!

Mac and iOS Forensic Analysis and Incident Response

POSTER

digital-forensics.sans.org

DFPS_FOR518_0924

Top Tip!

Parse NSKeyedArchiver plists using Deserializer:

https://github.com/ydkhatri/MacForensics/blob/master/Deserializer/deserializer.py

Native Apple Applications

Apple Mail – com.apple.mail/ com.apple.mobilemail

Description
This is a default email application that can be configured to use a number of email clients.

Location
macOS:
- Mailboxes: **~/Library/Mail/V#/<UUID>/*.mbox**
- Mailbox organization: **~/Library/Mail/V#/<UUID>/.mboxCache.plist**
- Envelope Index: **~/Library/Mail/V#/MailData/Envelope Index**

Downloaded email attachments may be stored in:
- **~/Library/Containers/com.apple.mail/Data/Library/MailDownloads/**
- **~/Downloads/**
- Extended Attributes (find using **ls -l@**)

iOS:
- **/private/var/mobile/Containers/Data/Application/<UUID>/**
- **/private/var/mobile/Library/Mail/**
- Envelope Index: **/private/var/mobile/Library/Mail/Envelope Index**

Interpretation
- The UUID folders can be correlated with the **Account4.sqlite** database.
- Mailboxes can contain nested mailboxes, messages and attachments.
  - **Messages** (macOS) or **MessageData** (iOS) folder contains raw EMLX email messages with an appended plist containing message metadata.
  - **Attachments** (macOS) or **AttachmentData** (iOS) folder contains message file attachments.
- Envelope Index sqlite database contains indexed mail data. It includes flags to show whether the email has been read, flagged, or deleted.
- Database timestamps are in Unix Epoch format.

Wallet and Apple Pay

Description
The Wallet application keeps track of tickets, cards, and passes. The user can add a credit card to the Apple Pay portion of the application to use for purchases.

Location
macOS:
- **~/Library/Passes/***
- iCloud synced data: **~/Library/Mobile Documents/com~apple~shoebox/UbiquitousCards/*.pkpass**

iOS:
- **/private/var/mobile/Library/Passes/***
- iCloud synced data: **/private/var/mobile/Library/Mobile Documents/com~apple~shoebox/UbiquitousCards/*.pkpass**

Interpretation
passes23.sqlite:
- Database timestamps are in Unix Epoch format.
- **UNIQUE_ID** will match the .pkpass filename.
- **GROUP_ORDER** shows the order of passes as shown to the user, with 0 at the top.
- Not all transactions may be saved in this database.
- Only Apple Card transactions are synced across devices.
- Transactions specific to Apple Cash are called "peer payments" and may have more related information.
- Journeys using a stored transit card are recorded, including start and end stations.

.pkpass files:
- Each card is a .pkpass package format directory.
  - **pass.json** stores the actual pass or card data.

Photos – com.apple.Photos

Description
Photos is the native photo gallery application, including photos and videos taken using the camera, screenshots, and synced media files.

Location
macOS:
- **~/Pictures/Photos Library.photoslibrary/***

iOS:
- **/private/var/mobile/Media/PhotoData/***
- **/private/var/mobile/Media/DCIM/***
- **/private/var/mobile/Media/PhotoStreamsData/***
- **/private/var/mobile/Media/PhotoCloudSharingData/***

Interpretation
- Photos **Library.photoslibrary** on macOS is a package format directory.
- Extended attributes show a file was synced from iCloud if **com.apple.quarantine** contains cloudphotosd.
- Photos application adds the **com.apple.assetsd.*** extended attribute.
  - This includes the original filename, location, timezone, flags for "hidden" and "favorite," and quarantine information.
- Photos taken with iOS 11+ use High Efficiency Image Container (HEIC) format.
- **Photos.sqlite** database includes metadata for each media file.
  - It includes extracted EXIF embedded metadata, annotations, location information, and detected faces and objects.
- Each subfolder of DCIM can contain up to 999 files, which are sequentially named from IMG_0001.
- Other photos may be stored by third-party applications – use **TCC.db** to find those with Camera permissions.

Reminders – com.apple.reminders

Description
Reminders are user-created lists that can consist of tasks to be completed, or those already marked as done.

Location
macOS:
- **~/Library/Reminders/Container_v1/Stores/***

iOS:
- **/private/var/mobile/Library/Reminders/Container_v1/Stores/***

Interpretation
- Each sqlite database contains reminders from a certain source (e.g., local, iCloud).
- Every object in the database has a different **Z_ENT** value, which changes for different versions of the database.
  - **OBJECT_TYPE** shows which type each **Z_ENT** refers to.

Calendar – com.apple.iCal/ com.apple.mobilecal

Description
This is the native calendar application on macOS and iOS with which items can be synced from a variety of accounts. It can include both personal and shared calendars.

Location
macOS:
- **~/Library/Calendars/***

iOS:
- **/private/var/mobile/Library/Calendar/***

Interpretation
- Each calendar directory contains an Events folder, which contains ICS calendar files and an **Info.plist** file.
- CalDAV **Info.plist** and ICS files may contain more information than those within the calendar directory.
- **Calendar.sqlitedb** contains calendar information.
  - Table names have changed over time.
  - It includes locations, shared events, notes, contacts, and more.
- Database timestamps are in Unix Epoch format in local time.

Messages – SMS and iMessage

Description
This is a native instant messaging application, which can be used with various different protocols.

Location
macOS:
- **~/Library/Messages/chat.db**
- **~/Library/Messages/Attachments/***

iOS:
- **/private/var/mobile/Library/SMS/sms.db**
- **/private/var/mobile/Library/SMS/Attachments/***

Interpretation
- SMS can only be used on iOS.
- Databases are sqlite and include messages and metadata.
  - Timestamps are in Mac Epoch format.
  - Apple Pay transactions are recorded in the **attributedBody** (BLOB) and **payload_data** fields (embedded binary plist).
  - **filename** field shows the path to an attachment.
  - **mime_type** shows the type of attachment.
  - **transfer_name** contains the attachment's filename.
  - Users can edit messages within 15 minutes, these edits are kept in the database.

Notes – com.apple.notes/ com.apple.mobilenotes

Description
Notes of various types can be created on macOS, iOS, and on iCloud.com. These can be local device notes, or synced to all devices associated with the same iCloud account.

Location
macOS:
- **~/Library/Group Containers/group.com.apple.notes/***

iOS FFS:
- **/private/var/mobile/Containers/Shared/AppGroup/<UUID>/***

Interpretation
- Even with syncing enabled, the user can choose to create local notes that are not synced.
- Note attachments are stored in the **Media** folder.
- Note thumbnails are stored in the **Preview** folder.
- Notes are stored in the sqlite database **NoteStore.sqlite**:
  - **Z_ENT** is an object type field. Different values for this field may have different meanings in different versions of the database.
  - **ZFILENAME** provides the attachment filename, as stored in the Media folder.
  - **ZISPASSWORDPROTECTED** = 0 (not encrypted), 1 (encrypted note).
  - **ZTITLE1** provides the note title.
  - **ZDATA** stores the note body as a BLOB, which is a protobuf in a GZIP archive.
  - **ZMARKEDFORDELETION** shows whether the note will be cleaned up at some point.

†**https://github.com/threeplanetssoftware/apple_cloud_notes_parser**

Contacts – com.apple.AddressBook

Description
The Contacts application (or Address Book) holds user contact information and can be populated by the user or by other applications.

Location
macOS:
- **~/Library/Application Support/AddressBook/***

iOS:
- **/private/var/mobile/Library/AddressBook/***

Interpretation
- Each source under the Sources folder could have its own associated database file and Metadata folder.
  - Metadata directories contain a binary plist file for each person (ending with p), subscription (s), or group (g).
  - Rename Metadata files to .plist in order to open with XCode.
- When searching for a person of interest, search for their UID, not just their name.
- Database timestamps are in Unix Epoch format in local time.

Contacts - FindMy

Description
The FindMy application allows people to share their location with others. If a user has any followers or contacts on FindMy, their metadata is stored in a JSON file.

Location
- **/private/var/mobile/Library/Caches/com.apple.findmy.fmfcore/FriendCacheData.data**

Call History – Phone and FaceTime

Description
Phone and FaceTime are the native calling applications on macOS and iOS.

Location
macOS:
- **~/Library/Application Support/CallHistoryDB/CallHistory.storedata**

iOS:
- **/private/var/mobile/Library/CallHistoryDB/CallHistory.storedata**

Interpretation
- Phone reverse DNS name is **com.apple.mobilephone**.
- FaceTime reverse DNS name is **com.apple.facetime**.
- Database is in sqlite format and includes calls made/received and metadata.
  - **ZDATE** timestamps are in Mac Epoch format in local time.
  - **ZADDRESS** = phone number or email address.
  - **ZANSWERED** = 0 (No), 1 (Yes).
  - **ZCALLTYPE** = 1 (telephony), 8 (FaceTime), 16 (FaceTime voice).
  - **ZORIGINATED** = 0 (incoming), 1 (outgoing with this user)
  - **ZDURATION** = time in seconds for this call.
  - **ZSERVICE_PROVIDER** = application used for the call.
- macOS database may store contact information in an encrypted BLOB.
- Some data may be synced across devices.

[iOS] Visual Voicemail

Description
Some, but not all, cellular carriers provide visual voicemail functionality on iOS devices, where voicemail audio files are downloaded to the device.

Location
- **/private/var/mobile/Library/Voicemail/***

Interpretation
- Each voicemail audio file (AMR file) has the **ROWID** from the voicemail.db sqlite database as a filename.
- If a voicemail has an accompanying transcript, this will be stored as an NSKeyedArchiver plist *.transcript file.
- Database timestamps are in Unix Epoch format in local time.

Maps – com.apple.Maps

Description
This is the native mapping application on macOS and iOS. Map data can be synced between devices using iCloud.

Location
macOS:
- **~/Library/Containers/com.apple.Maps/MapsSync_0.0.1**

iOS:
- **/private/var/mobile/Containers/Shared/AppGroup/CCB7770F-CF85-4292-8389-66232373192D/Maps/MapsSync_0.0.1**

Interpretation
- iOS backup folder for Maps may be empty.

MapsSync_0.0.1:
- **ZFAVORITEITEM** table contains the user's favorite locations
- **ZTYPE** = 2 (custom location)
- **ZHISTORYITEM** table shows user searches and queries (ZQUERY)
- **MIXINMAPITEM** table links together the history and favorite items
- **ZMAPITEMSTORAGE** stores location data as an embedded protobuf.

[iOS] Health

Description
Health information about the user is stored in a database, if enabled. This can include steps, distance, and heart rate, which can be collected using the Apple Watch.

Location
- **/private/var/mobile/Library/Health/healthdb_secure.sqlite**

Interpretation
- Database is encrypted in iOS backups, but not in a Full Filesystem dump.
- Use APOLLO† **health_*** modules to extract a user's health data.

†**https://github.com/mac4n6/APOLLO**

Contact Interactions – InteractionC.db

Description
This database keeps track of who the user is communicating with using applications such as Messages, Mail, and Phone.

Location
macOS:
- **/private/var/db/CoreDuet/People/interactionC.db**

iOS:
- **/private/var/mobile/Library/CoreDuet/People/interactionC.db**

Interpretation
- Data stored in this database includes the direction of communication (**INCOMING** or **OUTGOING**), start and end Mac Epoch timestamps, the application bundle id, contact information, and sender and recipient information.

[iOS] CarPlay

Description
Some information about vehicles the iOS device has been paired with are stored in plist files.

Location
- **/private/var/mobile/Library/SpringBoard/<UUID>-CarDisplayIconState.plist**
- **/private/var/mobile/Library/Preferences/[com.apple.CarPlayApp|com.apple.carplay].plist**

Interpretation
- **<UUID>-CarDisplayIconState.plist** shows the organization of icons on connected vehicle screens. The connected vehicle description is stored under the metadata key.
- **com.apple.carplay.plist** shows the most current Icon State UUID and paired vehicle type
- **com.apple.CarPlayApp.plist** shows recent apps under **CARRecentAppHistory**

Network Information

Network Interfaces

Description
These are the network interfaces on the system, interface types, and Mac addresses.

Location
macOS:
- **/Library/Preferences/SystemConfiguration/NetworkInterfaces.plist**
- **/Library/Preferences/SystemConfiguration/preferences.plist**

iOS:
- **/private/var/preferences/SystemConfiguration/NetworkInterfaces.plist**
- **/private/var/preferences/SystemConfiguration/preferences.plist**

Interpretation
- Each interface has an Item key in **NetworkInterfaces.plist**
  - **SCNetworkInterfaceType** is **IEEE802.11** for wireless interfaces, and **Ethernet** for wired interfaces.
  - **IOMacAddress** contains the unique MAC address for the interface
  - It also contains the device model.

DHCP Settings

Description
Files in this folder contain the last known network settings for those interfaces using DHCP.

Location
- **/private/var/db/dhcpclient/leases/**

Interpretation
- Each file within this directory includes lease information, router MAC address, IP address, and SSID, for a specified interface.

Wireless Network Connections

Description
This lists connections to access points, including wireless settings. It includes access points added using the WiFi menu and those synced from another device.

Location
macOS:
- **/Library/Preferences/com.apple.wifi.known-networks.plist**

iOS:
- **/private/var/preferences/com.apple.wifi.known-networks.plist**

Interpretation
com.apple.wifi.known-networks.plist:
- **AddReason** key shows whether the data has been synced.
- **AddedAt** timestamp shows when the access point was added to this plist file.
- **JoinedByUserAt** provides the timestamp when the user specifically joined the access point.
- **JoinedBySystemAt** provides the timestamp when the system auto-connected to the access point.
- **LocationLatitude** and **LocationLongitude** provide the coordinates of the access point.
- **PersonalHotspot** shows whether this device connected via another device's hotspot.

Network Usage – Logs

Description
Unified logs and system logs include entries for network connections made on the system.

Location
- **Unified Logs**
- **System log**

Interpretation
- Search for sender "IPConfiguration" and where the log message contains "Lease" or "network changed".
  - Use **log show –info –predicate 'senderImagePath contains[cd] "IPConfiguration" and (eventMessage contains[cd] "SSID" or eventMessage contains[cd] "Lease" or eventMessage contains[cd] "network changed")'**
- Search logs for "configd", "SSID", or "en0" for a more detailed view of wireless activity.
- Search logs for "country code" to show the country codes associated with wireless access point connections.
  - Default code is "X0" when one is not available.

File and Folder Sharing

[macOS] Extended Attributes

Description
A few extended attributes can reveal file sharing, including the sender, recipient, and application used.

AirDrop allows users to "drop" files to another user's device if that device is close by and using WiFi or Bluetooth. Extended attributes for a received file will show the name of the device the file was sent from using AirDrop.

Location
Everywhere! See extended attribute names for files:
- **ls -l@**
  - **com.apple.quarantine** contains the service of sharingd
  - **com.apple.assetsd** attributes contain AirDrop data such as file metadata and whether it was trashed or hidden
  - **com.apple.metadata:*** contains Spotlight metadata
  - **com.apple.metadata:kMDItemWhereFroms** provides the time and application, (e.g., "Received via Messages file transfer")
    - For files shared via AirDrop, this attribute provides the name of the device the item was sent from.

View extended attributes for a file:
- **xattr -xl <file>**

Interpretation
- It shows files shared using AirDrop, email, Messages, and other applications.
- Spotlight database can be searched for these attributes to look for evidence of file sharing.
- Be aware that a device name can be changed by the user.

AirDrop ID & Discoverability

Description
AirDrop uses a StreamID to identify itself to other devices. The last StreamID used can be found in these plist files.

Location
macOS:
- **~/Library/Preferences/[ByHost]/com.apple.sharingd.<HWUUID>.plist**
- **~/Library/Preferences/com.apple.sharingd.plist**

iOS:
- **/private/var/mobile/Library/Preferences/com.apple.sharingd.plist**

Interpretation
- **StreamID** stores the device identifier.
- If an Apple Watch is being used to unlock a macOS device, this information can also be found in here.

AirDrop Activity – Unified Logs

Description
Files sent and received via AirDrop are tracked in Unified Logs. This includes a unique identifier for the transaction (AirDrop ID), the type of file being sent, whether the connection was accepted, and potentially where the received file ended up.

Location
- Unified Logs

Interpretation
- If you can analyze both the sending and receiving devices, you can tie the activity together using the AirDrop ID (ReceiverID). If only one device is available, attribution is much more difficult.
- Be aware that device hostnames can easily be changed.
- Log shows whether the connection was "Accepted" or "Declined."

Spotlight

Description
Spotlight indexes the system to allow the user to search for files quickly. Indexing includes metadata that indicates file sharing.

Location
- **~/Library/Application Support/com.apple.spotlight/com.apple.spotlight.Shortcuts.v3**
- **/.Spotlight-V100/Store-V2/<GUID>**
- **~/Library/Metadata/CoreSpotlight/index.spotlightV3**

Interpretation
- Presence of the **kMDItemOriginMessageID** attribute shows files received via Apple Mail.
- **kMDItemUserShare*** keys can show sender and recipient metadata for Messages or AirDrop file transfers.
- **kMDItemOriginApplicationIdentifier** shows the application used.

[macOS] Shared Folders

Description
Information and metadata for shared folders on the system

Location
- **/private/var/db/com.apple.xpc.launchd/disabled*.plist**
- **/Library/Preferences/com.apple.RemoteManagement.plist**
- **/private/var/db/dslocal/nodes/Default/sharepoints/*.plist**
  - List of shared folders and their metadata

Interpretation
disabled.plist:
- By default, none of these settings are enabled.
- Look for **com.apple.smbd** and/or **com.apple.AppleFileServer** as the bundle IDs for shared folders.

sharepoints/*.plist:
- Each shared folder has its own plist file.

iCloud Documents

Description
iCloud stores local copies of documents shared using various applications.

Location
macOS:
- **~/Library/Mobile Documents/**

iOS:
- **/private/var/mobile/Library/Mobile Documents/**

Interpretation
- Each subdirectory corresponds to an application and is named in reverse DNS format but using tildes (~).
- Extended attributes for these documents include the iCloud Person ID in **com.apple.ubd.prsid**.
- Hidden ***.icloud** files correspond to files that have not been downloaded to this device.
  - These are binary plist files that contain the file's name and size.

Program Execution/ Application Usage

Terminal History – Executed Commands

Description
Each user account stores a list of commands run in a zsh shell terminal within a hidden file in their home folder.

Location
- **~/.zsh_history**
- **~/.zsh_sessions/<GUID>.history**

Interpretation
- These are Plaintext files containing up to 1000 commands run in order of execution.
- An upgraded macOS system may also store up to the last 500 commands run in a bash shell, in:
  - **~/bash_history**
  - **~/.bash_sessions/<GUID>.history**
- The files are created the first time the Terminal application is run.
- History files are not updated until the user account logs out. Session files are updated when the Terminal is exited.
- Files can be viewed on a live system using the **history** command.
- **<GUID>.history** files contain commands executed in that session.

Screen Time

Description
This tracks time spent in applications, notifications, and device pickups by the user following a notification.

Location
macOS:
- **/var/folders/<darwin_user_dir>/0/com.apple.ScreenTimeAgent/RMAdminStore-*.sqlite**

iOS:
- **/private/var/mobile/Library/Application Support/com.apple.remotemanagementd/RMAdminStore-*.sqlite**

Interpretation
- Data is organized by hour and category.
- Data retention is ~three weeks on iOS, and ~five weeks on macOS.
- Device name and UUID are shown for synced data.

Application Usage – KnowledgeC

Description
Amongst other things, the KnowledgeC database tracks application usage, including start and end times, and how the application was launched and used.

Location
macOS:
- **~/Library/Application Support/Knowledge/knowledgeC.db**
- **/private/var/db/CoreDuet/Knowledge/knowledgeC.db**

iOS:
- **/private/var/mobile/Library/CoreDuet/knowedgeC.db**

Interpretation
- It stores approximately four weeks of data.
- **/app/usage** stream keeps track of applications being used and when.
  - Use APOLLO† **knowledge_app_usage** module.
- **/app/intents** stream keeps track of what is happening in applications.
  - Use APOLLO† **knowledge_app_intents** module.
- **/app/activity** stream keeps track of application activities and interactions.
  - Use APOLLO† **knowledge_app_activity** module.
- **/device/NowPlaying** stream keeps track of what is playing and how it is playing.
  - Use APOLLO† **knowledge_audio_media_nowplaying** module.

†**https://github.com/mac4n6/APOLLO**

Application Usage – Biomes

Description
Many items that were once stored in knowledgeC.db can now be found in Biome files. This data type is scattered across the filesystem and appears to be where more of this data will be stored in the future.

Location
macOS:
- **~/Library/Biome**
- **/private/var/db/biome/**

iOS:
- **/private/var/mobile/Library/Biome**
- **/private/var/db/biome/**

Interpretation
- Search for **App.InFocus** streams in Biome data.
- Each Biome record contains a protobuf file, which contains start and end times, the app bundle ID, and the transition reason.
- Each Biome record is stored in a flat file system, one after another.
- Biome records have somewhat different formats across devices and/or OS versions.
- Biome **sync.db** database keeps track of what devices are syncing across each other.

Application Usage – CurrentPowerlog

Description
This shows nearly the same data as knowledgeC.db and Biomes. However, it may include additional data such as CarPlay, home screen, lock screen, and Siri usage.

Location
macOS:
- **/private/var/db/powerlog/Library/BatteryLife/CurrentPowerlog.PLSQL**

iOS FFS:
- **/private/var/Containers/Shared/SystemGroup/<GUID>/Library/BatteryLife/CurrentPowerlog.PLSQL**

iOS Sysdiagnose:
- **/logs/powerlogs/CurrentPowerlog.PLSQL**

Interpretation
- It stores approximately three days of data.

Be wary of timestamps in this log – some, but not all, have an offset.

- Use APOLLO† **powerlog_app_usage** module to extract the correct application usage times.
- Use APOLLO† **powerlog_incallservice** module to extract call logs.
- Use APOLLO† **powerlog_camera_state** module to extract camera state information.
- Use APOLLO† **powerlog_app_frontmost** module to show which application is the "frontmost" app.
- Archived databases may also be present in the **Archives** directory

†**https://github.com/mac4n6/APOLLO**

Application Data

Application Data

Description
This determines application information, including name and version.

Location
macOS:
- **/Applications/<Bundle ID>/***
- **~/Library/Containers/.../<Bundle ID>/***
- **~/Library/Application Support/<Bundle ID>/***
- **~/Library/Preferences/*.plist**
- **~/Library/Caches/***

Sandboxed applications:
- **~/Library/[Group] Containers/<Bundle ID>/Data/Library/Application Support/<App Name>/**
- **~/Library/[Group] Containers/<Bundle ID>/Data/Library/Preferences**
  - **<TLD>.<Company>.<Application>.plist** file contains the user's preferences
- **~/Library/Caches**

Non-sandboxed applications (legacy):
- **~/Library/Application Support**
- **~/Library/Preferences**
  - **<TLD>.<Company>.<Application>.plist** file contains the user's preferences
- **~/Library/Caches**

iOS FFS:
- **/private/var/mobile/Containers/.../<Bundle ID>/***
- **/private/var/mobile/Containers/Bundle/Application/<GUID>/***
  - Contains application binary file
- **/private/var/mobile/Containers/Data/Application/<GUID>/***
- **/private/var/mobile/Containers/Shared/AppGroup/<GUID>/***
  - Data shared among apps with the same developer
- **/private/var/mobile/Library/Cache/<Bundle ID>/***
- **/private/var/mobile/Preferences/*.plist**

Interpretation
- Each container is named in reverse DNS format.
- Each container directory contains a **.com.apple.containermanagerd.metadata.plist** file with application information.
- Each container directory contains a **Data** directory. The most interesting subdirectories are likely those that are not links.
- **Info.plist** file contains app name, bundle ID, and version information.
  - Similar information for iOS apps is stored in iTunesMetadata.plist, including the purchase date and user information.
- Cached items rarely get backed up, so typically won't be found in an iOS backup.

Keyboard Dictionary

Description
When a user types words into the device's keyboard, certain words are recorded in user dictionary files to help with autocorrection and predictive text features. These files should not contain anything typed into sensitive fields such as passwords, although may include sensitive data the user may have typed into non-secure areas such as notes. These files may or may not be included in iOS backups.

Location
macOS:
- **~/Library/Spelling/*dynamic-*.dat**

iOS:
- **/private/var/mobile/Library/Keyboard/*dynamic-*.dat**

Interpretation
- English dictionaries are dynamic-*.dat
- Other languages have their own files and will be preceded by their language abbreviation (e.g., ar for Arabic)

[iOS] Application Snapshots

Description
When an application is minimized to the background, a screenshot of the current screen is saved to the filesystem, to be used as a preview. App developers can choose to replace the screenshot with another image. This is commonly done for security reasons, such as in banking applications.

Location
- **<Application directory>/Library/Splashboard/Snapshots/<Bundle ID>/***

Interpretation
- Snapshots are PNG files
- They are usually only available on FFS acquisitions
- Previous versions of iOS generated snapshots as KTX files

Autorun Applications

Description
Autorun applications are those that automatically run when a user logs in.

Location
macOS:
- **/System/Library/LaunchAgents/*.plist**
- **/Library/LaunchAgents/*.plist**
- **~/Library/LaunchAgents/*.plist**
- **/System/Library/LaunchDaemons/**
- **/Library/LaunchDaemons/**

macOS 14:
- **/private/var/db/com.apple.backgroundtaskmanagement/BackgroundItems-v9.btm**

macOS 13:
- **~/Library/Application Support/com.apple.backgroundtaskmanagementagent/backgrounditems.btm**

iOS:
- **/Library/LaunchAgents/*.plist**
- **/System/Library/LaunchDaemons/**
- **/System/Library/NanoLaunchDaemons/**
- **/Library/LaunchDaemons/**
  - Requires jailbroken device to acquire

Interpretation
- Login items can be hidden from view of the user.
- Launch Daemons are background system processes.
- Launch Agents are background user processes.
- plist files are named in reverse DNS format.
- **BackgroundItems-v9.btm** / **backgrounditems.btm** is an NSKeyedArchiver plist file.

Saved Application State

Description
Application State information is stored, to allow it to be returned to its previous state after a reboot, if the user selects "reopen windows when logging back in" on shutdown.

Location
macOS:
- macOS legacy apps: **~/Library/Saved Application State/<bundle_id>.savedState/**
- macOS sandboxed apps: **~/Library/Containers/<Bundle ID>/Data/Library/Application Support/<App Name>/Saved Application State/<bundle_id>.savedState/**

iOS:
- **<Application directory>/Library/Saved Application State/<bundle_id>.savedState/**

Interpretation
- The existence of these directories indicates the user has used these applications.
- Each *.savedState directory contains at least two files:
  - [macOS] **windows.plist** and **data.data**
  - [iOS] **restorationinfo.plist** and **data.data**

Application Notifications

Description
Notifications for applications are stored by the Graphical User Interface for the system. For macOS, this is called Finder; for iOS, it is SpringBoard.

Location
macOS:
- **/private/var/folders/<DARWIN_USER_DIR>/com.apple.notificationcenter/db2/db**

iOS:
- **/private/var/mobile/Library/UserNotifications/<app GUID>/*.plist**

Interpretation
macOS:
- The user's **DARWIN_USER_DIR** path will be different for each user on the system.
- Attachments to notifications will be found in the **/attachments** directory.
- Database tracks notification delivery date, app bundle IDs, presentation, and style.
  - **NOTIFICATION DATA** is a BLOB that contains a binary plist.

iOS:
- plist files are in NSKeyedArchiver format.
- Notifications cleared by the user are removed from plist files.
- Other files in the same folder contain interface-specific items such as background pictures, icon layouts, and widgets.
- Pair app GUIDs with their associated bundle IDs by looking in **[/private/var]/mobile/Library/UserNotificationsServer/Library.plist**. This file is not included in iOS backups.

Software Installation and Updates

Description
This determines installed applications and updates, including timestamps, package names, and software used to install an application.

Location
macOS:
- **~/Library/Caches/com.apple.appstoreagent/storeSystem.db**
- **/Library/Receipts/InstallHistory.plist**
- **/Library/Preferences/com.apple.SoftwareUpdate.plist**
  - When system last checked for updates, how many updates were available, recommended updates.
- **/var/log/install.log**
  - Search the file for "Installed" to find app names and versions.
- **/var/db/receipts/**
  - Each software package install has a .bom and a .plist file.
  - plist file contains install timestamp, package name, and installer process.
  - bom file contains list of files and metadata for application.

iOS:
- **/private/var/installd/Library/Logs/MobileInstallation/mobile_installation.log.#**
  - Search the file for "Installing" to find app names and versions, for approximately one month of app installs.
  - Search for "Make container live" for app installs.
  - Search for "Destroying container" for app uninstalls.
  - Search app bundle IDs for specific app activity.
- **/private/var/mobile/Library/FrontBoard/applicationState.db**
  - It contains an embedded plist.

Interpretation
- **InstallHistory.plist** processName:
  - **macOS Installer** = System OS installer/updater
  - **softwareupdated** or "Software Update" = System/security updates
  - **storedownloadd** = App Store install
  - **Installer** = External installer
- bom file can be viewed using **lsbom <bom file>** command.
- **install.log** file will not include software installed via a drag and drop method.

Application Permissions – TCC

Description
Applications on macOS and iOS ask users which permissions they may have for different capabilities on the system. This is recorded in Transparency, Consent, Control (TCC) databases.

Location
macOS:
- **~/Library/Application Support/com.apple.TCC/TCC.db**
- **/Library/Application Support/com.apple.TCC/TCC.db**

iOS:
- **/private/var/mobile/Library/TCC/TCC.db**

Interpretation
- Apps may have access to permissions such as: Location, Contacts, Calendars, Photos, Bluetooth, Microphone, Camera, and Health
- **last_modified** = when this permission for this app was last updated in TCC
- **auth_value** = 0 (not allowed), 2 (allowed).
- **kTCCServiceUbiquity** permission is associated with iCloud.
- **kTCCServiceEndpointSecurityClient** permission is associated with Endpoint Security
- **KTCCServiceSystemPolicyAllFiles** permission is associated with Full Disk Access
- Not all permissions are shown to the user in the user interface.
- iOS TCC database is available in backup, physical and sysdiagnose acquisitions.

Third-Party Kernel Extensions

Description
Kernel modules are often used as device drivers, network filters, or support for filesystems, and can be used maliciously.

Location
- **/private/var/db/loadedkextmt.plist**
- **/Library/Apple/System/Library/Extensions/**
- **/System/Library/Extensions/**
- **/Library/Extensions/**
- **/Library/StagedExtensions/**
- **/Library/SystemExtensions/**
- **/Library/<Filesystems/macfuse.fs/Contents>/Extensions/**

Interpretation
- On a live system, use **systemextensionsctl list** command to list loaded system extensions and **kmutil showloaded** command to list loaded kernel extensions.
- Each extension is a bundle containing an **Info.plist** file.

Connected/Paired Devices and Backups

[macOS, Windows] Lockdown Files

Description
Connecting an iOS device to another system generates a lockdown file when the user selects "Trust This Computer".

Location
macOS:
- **/private/var/db/lockdown/**

Windows:
- **C:\ProgramData\Apple\Lockdown\**

Interpretation
- **<iDevice UDID>.plist** files are created for each iDevice paired with the system. Contains certificates, keybags, and other info used to access a locked device.
- Device PIN/passcode is required for pairing record creation.
- Lockdown records expire after 30 days of no use.

[macOS] Time Machine Backups

Description
Time Machine is the native backup utility on macOS, which may or may not be enabled.

Location
Time Machine settings:
- **/Library/Preferences/com.apple.TimeMachine.plist**
  - **BackupAlias** contains details about any backup disks.
  - **SnapshotDates** provide timestamps associated with backups.
  - It also includes other info such as filesystem type, encryption status, and backup frequency.

Logs:
- Unified logs
  - Use **log show –info –predicate 'process = "backupd"'** to show backup info

Interpretation
Unified logs show when the backup started and finished, network or local location of backup, volume name backed up, amount of data backed up, and deletion of old backups.

[macOS, Windows] iOS Backups

Description
iOS devices can be backed up to iCloud or to a local macOS or Windows system, either automatically or manually. Backups can be encrypted if the user chooses to enable this feature and set a backup password.

Location
macOS:
- **~/Library/Application Support/MobileSync/Backup/**

Windows XP:
- **C:\Documents and Settings\<user>\Application Data\Apple Computer\MobileSync\Backup\**

Windows Vista+:
- **C:\Users\<user>\AppData\Roaming\Apple Computer\MobileSync\Backup\**

Microsoft Store version of iTunes on Windows:
- **C:\Users\<user>\Apple\MobileSync\Backup\**

Interpretation
- Each subfolder is named for the device's UDID. A11-: 40-character UDID, A12+: [8 digits]-[16 digits] UDID.
- Folders named **<UDID>-<timestamp>** may also exist, which are created during a restore/update of the iDevice.
- **Status.plist** includes timestamps of the backup, type, and whether a full backup was performed.
- **Info.plist** contains device name, serial number, ICCID, MEID, IMEI, UDID, phone number, make, model, iOS and build information, the last backup date, and installed applications.
- **Manifest.plist** contains the backup date, whether the backup is encrypted, whether a device passcode was set, and the lockdown key, including device info, serial number, and UDID.
- iOS 10+: **Manifest.db** contains metadata about backup files. Previous versions of iOS stored this same data in Manifest.mbdb.
- A backup needs to be normalized by mapping files back to their original names. This may be shown differently by various tools.

[macOS] Attached iDevices – com.apple.iPod.plist

Description
All iDevices that have been attached to the system while logged in as that user are recorded in a plist file.

Location
- **~/Library/Preferences/com.apple.iPod.plist**

Interpretation
Devices' key contains one subkey per device, which includes the device type, IMEI, MEID, number of connections, time of last connection, and iOS version when last connected.

Bluetooth Devices

Description
Both macOS and iOS keep lists of Bluetooth devices that have been connected to the system.

Location
macOS:
- **/Library/Preferences/com.apple.MobileBluetooth.devices.plist**
- **/Library/Databases/com.apple.MobileBluetooth.ledevices.other.plist**
- **/Library/Preferences/com.apple.MobileBluetooth.ledevices.paired.plist**
- **~/Library/Application Support/Knowledge/knowledgeC.db**

iOS:
- **/private/var/containers/Shared/SystemGroup/<GUID>/com.apple.MobileBluetooth.devices.plist**
- **/private/var/containers/Shared/SystemGroup/<GUID/com.apple.MobileBluetooth.ledevices.other.plist**
- **/private/var/containers/Shared/SystemGroup/<GUID/com.apple.MobileBluetooth.ledevices.paired.plist**
- **/private/var/mobile/Library/CoreDuet/knowedgeC.db**

Interpretation
- Use timestamps carefully – certain user interactions can change how these timestamps may be interpreted. For example, changing the name of a device might update the first connected timestamp.
- **UnlockEnabled** = yes – an Apple Watch can be used to unlock this macOS device.
- **com.apple.MobileBluetooth.devices.plist** keeps track of connected Bluetooth devices.
  - Timestamps are stored in localtime.
- **com.apple.MobileBluetooth.ledevices.other.plist** file keeps track of "seen" Bluetooth low energy devices, that have not necessarily connected to the system.
  - Note that this file will not include all nearby Bluetooth-enabled devices.
  - Some device MAC addresses may be randomized.
  - **LastSeen** is Unix Epoch timestamp in local system time when this device was last used
- **com.apple.MobileBluetooth.ledevices.paired.plist** keeps track of paired Bluetooth low-energy devices.
- **KnowledgeC.db** keeps track of connected Bluetooth devices using the **/Bluetooth/isConnected** stream.
  - Use APOLLO† **knowledge_audio_bluetooth_connected** module.

FindMy – AirTags

Description
The FindMy application tracks information about the user's AirTags.

Location
- **/private/var/mobile/Library/Caches/com.apple.findmy.fmipcore/Items.data**

Interpretation
Items.data JSON file includes the owner, serial number, last connected timestamp and location information.

Deleted File or File Knowledge

Search – Spotlight

Description
Spotlight indexes the system to allow the user to search for files quickly. Indexing includes file metadata, extended attributes, and content of some file types.

Location
User shortcuts (searches):
- **~/Library/Application Support/com.apple.spotlight/com.apple.spotlight.Shortcuts.v3**
- **~/Library/Group Containers/group.com.apple.spotlight/com.apple.spotlight.Shortcuts.v3**

Main Spotlight indexing databases:
- **/.Spotlight-V100/Store-V2/<GUID>**
  - **VolumeConfiguration.plist** contains indexing exclusions and other configuration data.
  - **Cache** directory contains subdirectories of text-based versions of original documents, each named for the file's inode.
  - **[.]store.db** are the index databases.

User database:
- **~/Library/Metadata/CoreSpotlight/index.spotlightV3**

Interpretation
- A volume can explicitly be marked to disable indexing by placing a hidden, empty file named **.metadata_never_index** in the root of the volume.
- Some locations are not indexed by default, including DMG files, CDs, DVDs, hidden files and system directories.
- User shortcut files provide words actually typed in by the user.

[macOS] Files Quarantined by XProtect AV

Description
Some applications implement file tagging, so XProtect can automatically quarantine downloaded files that are deemed to be potentially malicious. Files that are quarantined are recorded in a database.

Location
- **~/Library/Preferences/com.apple.LaunchServices.QuarantineEvents.V2**

XProtect signature file:
- **/Library/Apple/System/Library/CoreServices/XProtect.bundle/Contents/Resources/Xprotect.plist**
  - **Xprotect.meta.plist** in the same folder contains the date the signature file was last updated.

Interpretation
- If an application is implementing this feature, it will have the **LSFileQuarantineEnabled** key set to True in its **Info.plist** file.
- Files copied off a USB or downloaded using an app that does not implement this feature will not be checked by XProtect.
- **LSQuarantineTimeStamp** = Timestamp when file was quarantined (Mac Absolute/WebKit time)
- **LSQuarantineAgentBundleIdentifier** = Application bundle ID that downloaded the file
- **LSQuarantineAgentName** = Application that downloaded the file
- **LSQuarantineDataURLString** = URL the file was downloaded from
- **LSQuarantineTypeNumber** = 0 (web browsers), 1 (XCode), 2 (Apple Mail), 3 (iChat), 6 (AirDrop), and 7 means another app.
- XProtect is only updated when Apple decides to update it and signatures are limited.
- The **com.apple.quarantine** extended attribute for a downloaded file may also contain useful information.

[macOS] Trash

Description
Any files or folders deleted by the user are saved into a hidden Trash folder in the root of that user's home directory.

Location
- **~/.Trash**

Interpretation
- Some trashed files can be restored using the "Put Back" option.
  - If the file has this option, the data can be found in the **.DS_Store** file in Trash.
- Safari "Safe" files are sent directly to Trash as they are auto-unzipped on download.
- Option available in **com.apple.finder.plist** to remove files from Trash after 30 days.
- iCloud may have its own Trash in the Mobile Documents directory.

File System Events Store Database

Description
Each volume connected to a Mac system will have a File System Events Store Database that is responsible for storing file system changes on the volume. It includes events such as file/folder creation and renaming, unzipping of files, item deletion, Trash being emptied, and volumes being mounted and unmounted.

Location
- **/.fseventsd/**

Interpretation
- Directory contains gzipped files that require root privileges to unzip and view.
- It can be wiped during a system crash or a hard power off.
- Events do not have associated timestamps. Approximate times can sometimes be estimated using filenames and paths.
- Parse using FSEvents Parser **https://github.com/mac4n6/FSEventsParser** (Updated fork of original script by Nicole Ibrahim)

Document Versions

Description
Document Versions (or Revisions) allows macOS to automatically back up certain types of documents or restore documents after a system crash. Versions are created when a document is saved, opened, every hour a document is open, and when it is frequently being edited. This feature is only supported by certain applications.

Location
macOS:
- **/System/Volume/Data/.DocumentRevisions-V100**

iOS FFS:
- **/private/var/.DocumentRevisions-V100**

Interpretation
- **/db-V1/db.sqlite** – Contains metadata for document versions
- **/.cs/ChunkStorage/*** – contains file versions
- Microsoft Office does not implement Document Versions; this has its own autosave feature.
- Users can access document versions within an application via File → Revert To → Browse All Versions...
- Hidden **.DocumentRevisions-V100** directory contains a folder named **PerUID** or **AllUIDs.**
  - Subdirectories are named <UID>, which are unique across all UIDs on system volumes.
- <UID> subdirectories contain further subdirectories named in reverse DNS format:
  - **com.apple.documentVersions** contains versions for documents saved on the local volume.
  - **com.apple.ubiquity** contains versions for documents saved on the local volume and iCloud.
  - **com.apple.thumbnails** contains versions for QuickLook thumbnails.
- Each file version or generation has extended attributes associated with "genstore."
  - **com.apple.genstore.info** contains an embedded binary plist that may include the hostname of the system on which the version was created.
  - **com.apple.genstore.origdisplayname** or **com.apple.genstore.posixname** stores the filename for this generation.
- Note that file versions may be shown as zero bytes in size.
- Some tracked files may not be stored using ChunkStorage, but instead stored inline in the APFS file system

To get a list of versions for a file:
- Find the file's inode number using **ls -li**
- Find entries for this inode in **db.sqlite** by joining the **GENERATIONS** and **FILES** tables

To get the content of a file version:
- Navigate to the **generation_path** provided in the **GENERATIONS** table in **db.sqlite**
- Run **ls -li** to get the file's inode
- Find the inode's associated entry in the **CSStorageChunkListTable** table in **ChunkStoreDatabase**
- Interpret the BLOB in **clt_chunkRowIDs** to extract entry ID(s)
- For each entry ID **ct_rowid** in the **CSStorageChunkListTable** table, get the filename stored in **ft_rowid**
- Each Chunk Storage data file is located in nested subdirectories of the /.cs folder and stores the file version's content after the 25th byte

Acquiring and Mounting Images

Mounting APFS E01 Image (With or Without FileVault)

Create mount point directories:
```
sudo mkdir /Volumes/apfs_image/
sudo mkdir /Volumes/apfs_mounted/
```
Create DMG file from E01 image:
```
sudo xmount --in ewf apfs.E01 --out dmg /Volumes/apfs_image/
```
Attach the image:
```
hdiutil attach -nomount /Volumes/apfs_image/apfs.dmg
```
List the disks to find the correct volume to mount:
```
(non-FileVault disk) diskutil ap list
(FileVault disk) diskutil ap unlockVolume <Disk GUID> -nomount
```
Mount volume:
```
sudo mount_apfs -o rdonly,noexec,noowners /dev/disk#
/Volumes/apfs_mounted/
```

Mounting APFS Snapshot

View APFS Snapshots available for system:
```
diskutil ap listsnapshots /System/Volumes/Data
```
Create mount point directory:
```
sudo mkdir /Volumes/snapshot_image/
```
Mount snapshot:
```
sudo mount_apfs -s snapshot.local
/System/Volumes/Data /Volumes/snapshot_mounted/
```

Mounting APFS DMG Image

Mount image:
```
hdiutil mount apfs.dmg -shadow
```
Using the -shadow option redirects writes to a file instead of modifying the original image.

Unmounting a Mounted Image

View mounted disks:
```
diskutil list
```
Eject mounted disk:
```
diskutil eject /dev/disk#
```
Find disk to unmount:
```
sudo mount
```
Unmount disk:
```
sudo umount /Volumes/disk_image/
```

Acquiring an Image of a Live System Using Apple System Restore (ASR)

Create a DMG as large as the disk is allocated:
```
hdiutil create -fs apfs -size <##GB> asrdisk.dmg
```
Make the new DMG available to the system:
```
sudo hdiutil attach -nomount asrdisk.dmg
```
Restore the source disk to the target DMG:
```
sudo asr restore --source /dev/disk# --target /dev/disk# --debug
--erase –verbose
```

Browser Usage and File Download

Safari Browser Session Restore

Description
Automatic Crash Recovery features are built into the browser.

Location
macOS:
- **~/Library/Containers/com.apple.Safari/Data/Library/Caches/com.apple.Safari/TabSnapshots/***
- **~/Library/Containers/com.apple.Safari/Data/Library/Caches/com.apple.Safari/TabSnapshots/Metadata.db**
  - Connects URL to the filename (UUID) in the TabSnapshots folder.

iOS:
- **/private/var/mobile/Library/Safari/BrowserState.db**
- **/private/var/mobile/Containers/Data/Application/<GUID>/Library/Safari/Thumbnails/*.png**

Interpretation
BrowserState.db:
- Visit timestamps are stored in Mac Epoch format.
- **order_index** shows the tab order.
- **private_browsing** shows 0 (regular) or 1 (private browsing) mode being used.
- **session_data** contains a BLOB.

Thumbnail KTX files:
- Each screenshot is a preview of a tab, including those in private browsing mode.
- Files only present for tabs open when Safari was last backgrounded.

Safari Browser History

Description
This is the history of websites a user has visited. Some may be synced from iCloud, if this setting has been enabled, with devices and synced URLs listed in the Cloud Tabs database.

Location
macOS:
- **~/Library/Safari/History.db**
- **~/Library/Containers/com.apple.Safari/Data/Library/Safari/SafariTabs.db**
- **~/Library/Containers/com.apple.Safari/Data/Library/Safari/CloudTabs.db**

iOS:
- **/private/var/mobile/Library/Safari/History.db**
- **/private/var/mobile/Library/Safari/SafariTabs.db**
- **/private/var/mobile/Library/Safari/CloudTabs.db**

Interpretation
History.db:
- On iOS, this data is retained for ~one month, on macOS, it's retained for ~one year by default (but can be re-configured).
- **history_items** contains URLS, domains, and visit counts.
- **history_visits** contains Mac Epoch timestamps of visits, and webpage titles.
- **Origin** = 0 (visit occurred on this device), 1 (synced from another system via iCloud).
- **SafariTabs.db** stores browser session data and currently open tabs, and Tab Groups (think Bookmarks).
- **CloudTabs.db** lists synced iCloud tabs and devices that are syncing the user's Safari web history.
- [macOS] **RecentlyClosedTabs.plist** also keeps track of recently closed tabs.

Safari Downloads

Description
Safari download history is stored in a configuration file and can indicate websites visited and items downloaded.

Location
- **~/Library/Safari/Downloads.plist**

Interpretation
- By default, items are removed from this list after one day.
  - This can be changed by the user to "When Safari Quits", "Upon Successful Download", or "Manually".
- **DownloadEntryURL** (macOS) and **sourceURL** (iOS) show where the download originated.
- **DownloadEntryPath** (macOS) is the file path to show where the item was downloaded to.
- **DownloadEntryDateAddedKey** (macOS) and **DateAdded** (iOS) indicate when the download started.
- **DownloadEntryDateFinishedKey** (macOS) and **DateFinished** (iOS) indicate when the download finished.

[macOS] Extended Attributes – File Download

Description
Apple uses extended attributes to store metadata about downloaded files, including the download date and where the file was downloaded from.

Location
Everywhere! See extended attribute names for files:
- **ls -l@**
  - **com.apple.quarantine** provides quarantine data for downloaded files, including download time (Unix Epoch hex) and application used to download the file.
  - **com.apple.metadata:kMDItemDownloadedDate** provides the download date in NSDate format (8-byte BE).
  - **com.apple.metadata:kMDItemWhereFroms** provides the URL the item was downloaded from, and referring URL.

View extended attributes for a file:
- **xattr -xl <file>**

Interpretation
- Not all applications will create all of the above extended attributes; attributes produced depend on the app developer.
- **kMDItemDownloadedDate** is not stored by Chrome
- **kMDItemWhereFroms** is not stored for Safari "Safe Files"

[macOS] Extended Attributes – Email Attachment Download

Description
A few extended attributes are created when an email attachment is downloaded.

Location
Everywhere! See extended attribute names for files:
- **ls -l@**
  - **com.apple.quarantine** provides the download time and application (e.g., Mail).

View extended attributes for a file:
- **xattr -xl <file>**

Safari Cookies

Description
Cookies provide insight into which websites have been visited and what activities might have taken place there.

Location
macOS:
- **~/Library/Safari/Cookies.binarycookies**
- **~/Library/Containers/com.apple.Safari/Cookies.binarycookies**

iOS:
- **/private/var/mobile/Library/Safari/Cookies.binarycookies**
- **/private/var/mobile/Containers/Data/Application/<UUID>/Library/Safari/Cookies.binarycookies**

Interpretation
- Safari also uses the Cookie file **com.apple.Safari.SafeBrowsing.binarycookie**
- Other applications that have internal browsers may have their own WebKit cache and cookies.

Safari Browser Cache

Description
Files may be cached by the browser when the user visits a webpage. Full reconstruction of a website using only cached items is unlikely.

Location
macOS:
- **~/Library/Containers/com.apple.Safari/Data/Library/Safari/com.apple.Safari/WebKitCache/Version ##/***

iOS:
- **/private/var/mobile/Containers/Data/Application/<UUID>/Library/Caches/WebKit/Version ##/***

Interpretation
- **Records/SubResources** folder contains a list of cached items per website visit and embedded SHA1 hashes for each file.
- **Records/Resources** folder contains cached data and metadata, including SHA1 of filename for related file in the **Blobs** folder.
- Additional cached data may exist in the **Blobs** folder.
- Cached data may be stored in multiple separate files if it's too large to fit into a single file. This is often seen with media.

macOS vs. Windows Artifacts*

plist files	⟷	Registry
fsevents	⟷	USNJrnl
DS_Store	⟷	Shellbags
Trash	⟷	Recycle Bin
Spotlight	⟷	Windows Search
Extended attributes	⟷	ADS
LoginItems & Launch Agents/Daemons	⟷	Autoruns
MRU	⟷	MRU
Spotlight	⟷	Prefetch
knowledgeC.db	⟷	SRUM

*NOTE: These are not exact like-for-like comparable artifacts, but do contain similar types of data.

macOS Artifacts on Non-Mac Systems

Copying data from a macOS system to a non-Mac system does not always copy everything.

	HFS+/APFS	FAT/exFAT
Document Versions	✔	✘
Spotlight	✔	✔
Trash	✔	✔
File System Events	✔	✔ (empty dir)
Extended Attributes	✔	✔ (separate (AppleDouble) file)
.DS_store files	✔	✔

File/Folder Opening

[macOS] Extended Attributes – DMG File Opened

Description
Double-clicking a DMG file produces two additional extended attributes for that file that are specific to this action and this file type. These extended attributes show that the DMG was opened at least once.

Location
Everywhere! See extended attribute names for files:
- **ls -l@**
  - **com.apple.diskimages.fsck** provides file system check information.
  - **com.apple.diskimages.recentcksum** provides checksum info and download date (Unix Epoch).

View extended attributes for a file:
- **xattr -xl <file>**

Interpretation
The first open timestamp from this process is recorded in **~/Library/Logs/fsck_hfs.log**

[macOS] Extended Attributes – File Last Used

BEWARE: This attribute may not show when the user last viewed a file as it's not always updated. For example: using the Finder QuickLook function will not update this timestamp.

Description
This extended attribute shows when a file was last viewed either using Finder or the "open" command in the Terminal.

Location
Everywhere! See extended attribute names for files:
- **ls -l@**
  - **com.apple.lastuseddate#PS** stores Unix Epoch timestamp of when file was last used, as it pertains to the file system

View extended attributes for a file:
- **xattr -xl <file>**

Interpretation
Not all file types have this attribute.

[macOS] .DS_Store – Folder Access

Description
Hidden DS_Store files can exist all over macOS systems, and are created when the Finder application is used to access a directory.

Location
Everywhere!
- **.DS_Store**

Interpretation
- These files implement a B-tree format.
- For trashed files, **.DS_Store** contains the original filename and original file path.

[macOS] Most Recently Used (MRU)

Description
A number of artifacts store information about recently accessed folders, applications, documents, hosts, and volumes on the system.

Location
- **~/Library/Preferences/com.apple.finder.plist**
  - **FXRecentFolders** key lists recently accessed folders in order, with the most recent under **Item 0**
  - **file-bookmark** BLOB contains the full folder path, Volume Name, and Volume GUID
- **~/Library/Application Support/com.apple.LSSharedFileList.*.sfl[2 or 3]**
  - Recent items per application, volume, or host

Microsoft Office 365:
- **~/Library/Containers/com.microsoft.<app>/Data/Library/Preferences/com.microsoft.<app>.securebookmarks.plist**
  - Each key includes the last-used timestamp in **kLastUsedDateKey**.
  - **kBookmarkDataKey** contains a bookmark data BLOB that includes the file path, volume name, and volume GUID.

Interpretation
- SFL files are binary plists that use the NSKeyedArchiver format.
- Most native MRU lists keep the last 10 items by default. Microsoft Office keeps more.

[macOS] Recent Folders

Description
These are folders recently accessed by the user account.

Location
- **~/Library/Preferences/com.apple.finder.plist**
  - **FXRecentFolders** contains a bookmark data BLOB in **file-bookmark**

Interpretation
- Item 0 is the most recent and item 9 is the least.

[macOS] Recent Items

Description
These are items recently accessed by the user account, per application.

Location
- **~/Library/Application Support/com.apple.LSSharedFileList.*.sfl[2 or 3]**

Interpretation
- The list contains both native and third-party applications.
- Files are named in reverse DNS format.

Volumes and External Device/ USB Usage

[macOS] Finder – Mounted Volumes

Description
The Finder application on macOS stores a list of volumes that have been mounted on the Desktop within a plist file. It includes the volume name with X and Y coordinates of volumes when mounted on the Desktop.

Location
- **~/Library/Preferences/com.apple.finder.plist**
  - **FXDesktopVolumesPositions** key

Interpretation
Note: Some volumes may have an appended number.†
- The key will not exist if the user does not have Finder preferences configured to show items on the Desktop.
- It includes host volumes, USB drives, and mounted DMG files.

† Darren Freestone has determined this is a "hexadecimal floating point constant" value representing the volume creation timestamp for HFS+/APFS volumes, but is only a negative value for FAT/ExFAT volumes.

[macOS] Logs – Mounted Volumes

Description
Logs record what volumes were mounted on the system and can include the device file the volume is using, volume size, name, and mount point.

Location
- **/var/log/daily.out**
- System log
- Unified logs

Interpretation
- Search for "**/Volumes/**" to find any volumes mounted under the default mount point.
- You can also search **system.log** and unified logs for **apfs**, **hfs**, **mounted**, **unmounted**, or **disk#s#**.
- Find connections to network shares by searching for **afp://**, or on older systems: **afpfs://**, **smbfs://**, or **smb://**
- Searching on the volume name can find activity relating to that volume.
- Daily logs record what volumes were mounted on the system when the daily maintenance script was run.

[macOS] Favorite Volumes

Description
These are a list of favorite volumes, including the volume name and properties.

Location
- **~/Library/Application Support/com.apple.sharedfilelist/com.apple.LSSharedFileList.FavoriteVolumes.sfl[2 or 3]**

Interpretation
- NSKeyedArchiver plist file containing Bookmark BLOBs.

[macOS] Search Logs for Connected USB Devices

Description
The USB Mass Storage Class (USBMSC) Identifier can be used to find USBMSC device connections in the System log and in Unified logs, including the device serial number, vendor, and product information.

Location
- System log
- Unified logs

Interpretation
- Search for **USBMSC**
  - Typical structure of these records: USBMSC Identifier (non-unique): <serial number> <VID> <PID> <version>
- Be aware that not all USBMSC entries are user-initiated.
- You can also find network share connections by filtering Unified Logs: **process = NetAuthSysAgent AND sender = loginsupport**

Account Usage

[macOS] com.apple.loginwindow.plist

Description
Last logged-in user, current logged-in user (on live system), auto-login user (if configured), and other settings are recorded in a plist file.

Location
- **/Library/Preferences/com.apple.loginwindow.plist**

Interpretation
- A user may choose "Automatic login" in preferences. Their (XOR'd) password is then stored in **/etc/kcpassword**. Decode using the script from **https://gist.github.com/opshope/32f65875d45215c3677d**
- Automatic login is not available for user FileVault or iCloud credential logins.

[macOS] User Logins

Description
These are successful and failed user account login and logout events.

Location
- System log
- Unified logs
- ASL

Interpretation
- Login events are marked with **USER_PROCESS** and the process ID.
- Login type is identified by:
  - **loginwindow** = login via the GUI
  - **login** = login via the Terminal
  - **sshd** = login via SSH
  - **screensharingd** = Screen Sharing
- Logoff events are marked with **DEAD_PROCESS** and the process ID.

[macOS] su Logins

Description
These are successful and failed su logins.

Location
- Audit logs
- Unified logs

Interpretation
- View su logins in Audit logs: **praudit -xn /var/audit/* - su**
- View attempts to use sudo in Unified Logs: **log show --predicate '(process == "su" or process == "sudo") and eventMessage contains "tty"'**

[macOS] Account Creation

Description
Entries in the audit log are added when a user account is created.

Location
- Audit logs

Interpretation
- **create user event** includes the name of the new user and the UID of the user who created.

[macOS] Screen Lock/Unlock

Description
Events are recorded when the screen is locked or unlocked.

Location
- Unified logs

Interpretation
- Screen lock events contain **com.apple.sessionagent.screenIsLocked**
- Screen unlock events contain **com.apple.sessionagent.screenIsUnlocked**
- This includes unlock actions using a regular password, TouchID, or Apple Watch.

[macOS] Known SSH Hosts

Description
These are Hostnames, IP addresses, and public keys for hosts that this system has connected to via SSH, for which the user decided to save the key.

Location
- **~/.ssh/known_hosts**
- **~/.ssh/authorized_hosts**

Interpretation
- By default, hostnames and IP addresses will be readable.
  - This data will be hashed if **HashKnownHosts** is set to yes in the **/etc/ssh/ssh_config** file.

[macOS] su Privilege Escalation

Description
Users with su privileges are recorded, as well as a log of commands that have been run as root.

Location
Users with root-level privileges:
- **/etc/sudoers**

Unified logs

Interpretation
- Look for the sudo or su process.

Physical Location

Applications Requesting Location Permissions

Description
The system records a list of applications that have requested location services.

Location
macOS:
- **~/Library/Application Support/com.apple.TCC/TCC.db**
- **/Library/Application Support/com.apple.TCC/TCC.db**
- **/private/var/db/locationd/clients.plist**

iOS:
- **/private/var/mobile/Library/TCC/TCC.db**
- **/private/var/root/Library/Caches/locationd/clients.plist**

Interpretation
TCC.db:
- It includes **last_modified** timestamp for each permission for each application.
- **auth_value** = 0 (not allowed), 2 (allowed).
- **KTCCServiceLiverpool** permission is generally assumed to be part of location services.
- iOS TCC database is available in backup, physical and sysdiagnose acquisitions.

clients.plist:
- **Authorization** = 1 (Never), 2 (While Using), 4 (Always),
  - No Authorization key means "Ask."
- **CorrectiveCompensationEnabled** = 1 (or no key) means Precise Location is enabled, 2 means disabled.

[iOS] Frequent and Significant Locations

Description
When enabled, the Significant Locations setting uses location services to keep track of a device's location and finds routines in their pattern.

Location
- **/private/var/mobile/Library/Caches/com.apple.routined/*.sqlite**

Interpretation
- Setting can be enabled or disabled in **Settings** → **Privacy** → **Location Services** → **System Services** → **Significant Locations**.
- Algorithm to establish how the device marks a location as "frequent" is unknown.
- **Cloud[-V2].sqlite** database shows visits to certain locations.
  - Use APOLLO† **routined_cloud_visit_entry** module to extract location visits.
- **Cache.sqlite** database contains very granular location data for about one week.
  - Use APOLLO† **routined_cache_zrtcllocationmo** module to extract location visits.
- Data is also found on macOS, however it is encrypted.

†**https://github.com/mac4n6/APOLLO**

Cellular and WiFi Locations

Description
Locations of various cellular and WiFi access are recorded in a few databases.

Location
macOS:
- **/private/var/folders/*/<DARWIN_USER_DIR>/cache_encrypted*.db**
- **/private/var/folders/*/<DARWIN_USER_DIR>/lockCache_encrypted*.db**

iOS FFS:
- **/private/var/root/Library/caches/locationd/cache_encrypted*.db**
- **/private/var/root/Library/caches/locationd/lockCache_encrypted*.db**

Interpretation
- Data is retained for ~one week, but this varies per table.
  - Data in the WifiLocation table is retained for ~four days.
- Timestamps are stored in Mac Epoch and appear to be accurate.
- Locations are accurate to within the general area.
- MAC addresses are stored in Base10.
- <DARWIN_USER_DIR> will be different for each user and is explained in more detail at: **http://www.swiftforensics.com/2017/04/the-mystery-of-varfolders-on-osx.html**
- Use APOLLO† **locationd_cacheencryptedAB_ltecelllocation** module to extract location data.

†**https://github.com/mac4n6/APOLLO**

FindMy – Device Location

Description
The FindMy application tracks a user's iCloud connected devices in a JSON file.

Location
- **/private/var/mobile/Library/Caches/com.apple.findmy.fmipcore/Devices.data**

Interpretation
- This can include devices such as AirPods.
- Includes last connected timestamp and location.

Log Files

Apple System Log (ASL)

Location
- **/private/var/log/asl/**
  - **YYYY.MM.DD.[UID].[GID].asl**
  - Login records (utmp, wtmp, lastlog): **BB.YYYY.MM.DD.[UID].[GID].asl**

Additional syslog data directories:
- **AUX.YYYY.MM.DD**

Interpretation
- View using Console.app or syslog command.
- Messages logged by syslog: TTL is seven days.
- Messages logged by utmp, wtmp, and lastlog: TTL is 366 days.
- Timestamps are stored in UTC.
- Collate logs: **syslog -F raw -T utc -d /private/var/log/asl/ > asl.log**
  - Open in Console: **open -a Console asl.log**

[macOS 13-] Audit Logs

Location
- **/private/var/audit/<start_time YYYYMMDDHHMMSS>.<end_time YYYYMMDDHHMMSS>**

Audit log configuration files:
- **/etc/security/audit_***

Interpretation
- Deprecated on macOS 11, disabled in macOS 14
- Timestamps are stored in UTC.
- **praudit** command may output timestamps in local time.
  - Use **TZ=UTC** command to temporarily change terminal timezone to UTC.
- Collate logs: **praudit -xn /private/var/audit/*.* > audit.log**
  - Open collected log in Console: **open -a Console audit.log**

System.log

Location
- **/private/var/log/system.log**

Interpretation
- Timestamps are stored in localtime.
- Concatenate system logs into one file using the command: **gzcat system.log.{6..0}.gz > system_all.log**

Unified Logs

Location
- **/private/var/db/diagnostics/*.tracev3**
- **/private/var/db/uuidtext/***
  - Messages associated with **SessionAgentNotificationCenter** show user-initiated actions relating to system shutdown events.

Interpretation
- Timestamps are stored in UTC.
- Create logarchive bundle for offline analysis:
  - Create logarchive folder: **sudo mkdir logs.logarchive**
  - Copy log files: **cp -R /private/var/db/uuidtext/ /private/var/db/diagnostics/ logs.logarchive**
  - Make logarchive format: **/usr/libexec/PlistBuddy -c "Add :OSArchiveVersion integer 4" logs.logarchive/Info.plist**
- Analysis:
  - Get USBMSC entries: **log show logs.logarchive/ --timezone UTC --info --predicate 'eventMessage contains "USBMSC"'**
  - Search for a device's volume name: **log show logs.logarchive/ --timezone UTC --info --predicate 'eventMessage contains "VOL_NAME"'**
  - Export unified logs to text file: **log show logs.logarchive/ --timezone UTC --info > galaga_logs.txt**
  - List shutdowns/reboots: **log show logs.logarchive/ --timezone UTC --info --predicate 'eventMessage contains "com.apple.system.loginwindow" and eventMessage contains "SessionAgentNotificationCenter"'**
  - List shutdown cause: **log show logs.logarchive/ --timezone UTC --info --predicate 'eventMessage contains[c] "shutdown cause"'**
  - Get backup logs: **log show logs.logarchive/ --timezone UTC --info --predicate 'process = "backupd" and category = "general"'**
  - Get network logs: **log show logs.logarchive/ --timezone UTC --info --predicate 'senderImagePath contains[cd] "IPConfiguration" and (eventMessage contains[cd] "SSID" or eventMessage contains[cd] "Lease" or eventMessage contains[cd] "network changed")'**

System and User Information

Operating System Version and Serial Number

Description
This determines the OS version, build version, and serial number.

Location
macOS:
- **/System/Library/CoreServices/SystemVersion.plist**
  - OS version, build version
- **/private/var/folders/*/<DARWIN_USER_DIR>/cache_encryptedA.db**
  - Serial number

iOS:
- **/mobile/Library/Logs/AppleSupport/general.log**
- **/logs/AppleSupport/general.log**
  - Device model, OS version, serial number
- **/private/var/containers/Data/System/<GUID>/Library/activation_records/activation_record.plist**
- **/private/var/containers/Data/System/<GUID>/Library/activation_records/wildcard_record.plist**
  - Device UDID, IMEI, model, serial number
- **/private/var/mobile/Library/Preferences/com.apple.springboard.plist**
  - Device locale, OS version, as well as settings such as erase device after 10 failed passcode attempts
- **/private/var/mobile/Library/Preferences/com.apple.purplebuddy.plist**
  - Device setup info, including original locale, setup time, and hardware model
- **Info.plist**
  - Device hostname, model, UDID, iOS version, serial number

Operating System Installation Date

Description
This determines the OS installation date and date of updates.

Location
- **/private/var/db/.AppleSetupDone**
  - Date of last OS update: stat -x /private/var/db/.AppleSetupDone (Change date).
- **/private/var/log/install.log**
  - OS installation date: grep "Installed\ \"macOS" install.log
- **/private/var/db/softwareupdate/journal.plist**
  - InstallDate keys show OS installation timestamps.
- **/private/var/mobile/Library/Preferences/com.apple.purplebuddy.plist**
  - Device setup info, original locale, setup time, device model.

Interpretation
There may be a difference in time zones – original time zone is Cupertino, before user sets their own.

User Accounts

Description
Each user and group has their own plist file.

Location
- **/private/var/db/dslocal/nodes/Default/users/**
- **/private/var/db/dslocal/nodes/Default/groups/**

Interpretation
Files may be binary or XML plist files depending on the OS version.
- Access to these directories requires root privileges.
- Each plist file contains the account creation timestamp, last password reset time, username, and potentially the associated email address.
- Timestamps are stored in Unix Epoch format.
- failedLoginCount and failedLoginTimestamp values do not appear to be updated.

[macOS] User Account Passwords

Description
User account password hashes are stored locally. The format and location of these has changed with different versions of macOS.

Location
- **/private/var/db/dslocal/nodes/Default/users/***
  - **ShadowHashData** key in plist files contains the password hash.

Interpretation
- Password hash is a salted SHA512 PBKDF2 hash.
- John The Ripper (JTR) and Hashcat include password cracking support for all of these hashes.

Deleted User Accounts

Description
If any user accounts have been deleted on the system, they will be listed in a plist file under the **deletedUsers** key. This file may not exist if no accounts have been deleted.

Location
- **/Library/Preferences/com.apple.preferences.accounts.plist**

Interpretation
- When a user account is deleted, the user's plist in the **/private/var/db/dslocal/nodes/Default/users/** directory is also removed.
- Lists user's name, UID, username, and deletion date for each account.
- Three options for the user's data are made available when an account is deleted:
  - Save the home folder to a DMG file, which is saved to **/Users/Deleted Users/**
  - Leave the home folder in place.
  - Remove the user's home directory.

Time Zone

Description
This determines the current time zone of the system.

Location
- **/etc/localtime**
- **/Library/Preferences/.GlobalPreferences.plist**

Interpretation
The .GlobalPreferences.plist file contains the time zone configuration data. It may not be updated when switching between static location and location services.
- **/Library/Preferences/com.apple.timezone.auto.plist** shows if location services are enabled.
- Timezone changes are recorded in **system.log** and Unified Logs.
  - Timestamps stored in localtime in system log and UTC in Unified Logs.
  - Search for "location" or "timezoned".
  - Timestamp jumps may also be visible in **/var/log/*** as these logs record events in local time.
  - Last modified timestamp of **/etc/localtime** symlink is updated when the timezone is changed.

[iOS] Evidence of Jailbreaking

Description
Some indicators may exist that point to a device being jailbroken. Indicators will differ depending on the device and type of jailbreak used.

Location
- **/private/etc/fstab**
  - Look for System partition mounted as rw.
- **/Applications**
  - Look for unofficial app stores associated with jailbreaks. Common apps: Cydia, Bydia, Zydia, Installer, 25pp, Maiyadi.
  - Look for apps associated with jailbreaks. Common apps: Meridian, LiberiOS, mac_portal, Pangu, unc0v3r, rootlessJB, checkra1n.
  - Look for unauthorized apps associated with jailbreaks. Common apps: iFile, SBSettings, or SSH, tethering, and configuration apps.
- Files or directories associated with any of the above apps, or forensic utilities (e.g., dumpkeys6 is created by Elcomsoft).

iCloud-Synced Accounts and Preferences

Description
Each iCloud account synced to the system will be recorded as a file named for the iCloud Person ID in the iCloud Accounts folder. This same directory contains links named for each email address associated with an iCloud account that points to the relevant iCloud Person ID for that account. Preferences are also synced across devices into the SyncedPreferences folder.

Location
macOS:
- **~/Library/Application Support/iCloud/Accounts/***
- **~/Library/SyncedPreferences/**
- **~/Library/Containers/<Bundle ID>/Data/Library/SyncedPreferences/**

iOS FFS:
- **/private/var/mobile/Containers/...**
- **/private/var/mobile/Library/SyncedPreferences/**

Interpretation
Each application syncing with iCloud has its own plist in the SyncedPreferences folder.

[macOS] Firewall Configuration

Description
The Application-Level Firewall (ALF) is turned off by default. It is one of two default firewalls on macOS systems. The second is the IP/packet filtering firewall.

Location
ALF configuration:
- **/Library/Preferences/com.apple.alf.plist**
  - **globalstate** = 1 (firewall enabled), 0 (firewall disabled).
  - **allowsignedenabled** = 1 (allow signed software to receive incoming connections).
  - **allowdownloadsignedenabled** = 1 (allow downloaded signed software to receive incoming connections).
  - **stealthenabled** = 1 (stealth mode enabled).
  - **applications** key lists apps configured in the firewall.
    - **state** = 0 (incoming connections allowed), 2 (incoming connections blocked).

Packet filter firewall configuration:
- **/etc/pf.conf**

Managed Device Profiles

Description
Devices can be managed through enterprise Mobile Device Management systems or settings pushed to the device by an organization or carrier. These devices have a configuration profile installed, which outlines allowed actions and limitations. Provisioning profiles allow apps to run without being downloaded from the App Store (sideloading).

Location
macOS:
- **/private/var/db/ConfigurationProfiles/**

iOS:
- Configuration profiles:
  - **/private/var/mobile/Library/UserConfigurationProfiles/**
  - **/private/var/containers/Shared/SystemGroup/systemgroup.com.apple.configurationprofiles**
- Provisioning profiles:
  - **/private/var/MobileDevice/ProvisioningProfiles/**

Interpretation
- Use "profiles" command to extract detailed configuration.
- Malware and jailbreaks can use provisioning profiles, as well as legitimate MDM solutions. Look for app names, timestamps, and developer certificates.
- Provisioning profile plist:
  - CreationDate key is when the app was sideloaded.
  - ExpirationDate will show to expire after seven days for a free developer account or 365 days for a paid account.
  - ProvisioningDevices key shows UDIDs for all devices that also have this application installed.

[iOS] Cellular Information

Description
Cellular information is information associated with the device and SIM. It includes the current and historical ICCID, phone number(s), IMSI, and carrier information.

Location
- **/private/var/wireless/Library/Preferences/com.apple.commcenter.data.plist**
- **/private/var/wireless/Library/Databases/CellularUsage.db**

Interpretation
- Timestamps may not necessarily reflect expected SIM usage.
- **CarrierBundleName** can be used to map carrier ID to name.
- Note the **esim** key is set to **1** when the device uses an eSIM instead of a physical SIM

[macOS] System Boot, Reboot, and Shutdown

Description
When the system boots up and is shut down is recorded within log files.

Location
- System log
  - Search for "BOOT_TIME" and "SHUTDOWN_TIME" – these entries include an associated Unix Epoch timestamp.
- Unified logs
  - Messages associated with **SessionAgentNotificationCenter** show user-initiated actions relating to system shutdown and restart events.

Interpretation
- Search for "halt" for shutdown events and "reboot" for reboot events.
- The system records the reason for the sleep/shutdown as "Sleep Cause" or "Shutdown Cause."
  - <0 = error
  - 0 = hibernation (sleep) or battery removal/power plug (shutdown)
  - 3 = hard shutdown (power button held)
  - 5 = normal sleep/shutdown

Device Lock/Unlock and Plugged In – KnowledgeC

Description
Amongst other things, the KnowledgeC database tracks when the device is locked or unlocked and when it is plugged in or power is disconnected.

Location
macOS:
- **~/Library/Application Support/Knowledge/knowledgeC.db**

iOS FFS:
- **/private/var/mobile/Library/CoreDuet/knowedgeC.db**

Interpretation
- Stores approximately four weeks of data
- **/device/isLocked** stream keeps track of when a device is locked and unlocked.
  - Use APOLLO† **knowledge_device_locked** module.
- **/device/isPluggedin** stream keeps track of power connection and disconnection events.
  - Use APOLLO† **knowledge_device_pluggedin** module.

Battery Levels – CurrentPowerlog

Description
CurrentPowerlog keeps track of the device's battery status and whether it is charging.

Location
macOS:
- **/private/var/db/powerlog/Library/BatteryLife/CurrentPowerlog.PLSQL**

iOS FFS:
- **/private/var/Containers/Shared/SystemGroup/<GUID>/Library/BatteryLife/CurrentPowerlog.PLSQL**

iOS Sysdiagnose:
- **/logs/powerlogs/CurrentPowerlog.PLSQL**

Interpretation
- It stores approximately three days of data.
- Be wary of timestamps in this log – some, but not all, have an offset.
- Archived databases may also be present in the **Archives** directory
- Use APOLLO† **powerlog_battery_level** module to extract battery information.

[macOS] Installed Printers and Print Jobs

Description
This shows the printers and scanners that are installed on the system and their configurations.

Location
- **/Library/Preferences/org.cups.printers.plist**
  - Each Item key refers to an installed printer.
- **/etc/cups/ppd/*.ppd**
  - One file per printer; contains capabilities such as page size, resolution, and color.
- **/private/var/spool/cups/c#####**
  - Print job control files containing metadata about a print job with ID corresponding to the filename.
  - Persistent files
- **/private/var/spool/cups/d#####-###**
  - Print job PDF data files are named in line with corresponding control file.
  - Non-persistent files should be removed immediately after the print job completes, unless job is cancelled or an error occurred.

Interpretation
- Clues in device-uri such as dnssd or tcp.local indicate a network-connected printer (as opposed to a cable).
- Print job control files include which printer was used, the originating user account, job name, and application used.

[macOS] Screen Sharing and Remote Login Preferences

Description
These are settings for items that can be shared, including screen sharing and remote access to the system.

Location
Preferences:
- **/private/var/db/com.apple.xpc.launchd/disabled*plist**
- **/Library/Preferences/com.apple.RemoteManagement.plist**
  - Created when screen sharing or remote management options are enabled.
- **/Library/Preferences/com.apple.VNCSettings.txt**
  - Contains the XOR'ed password to access the system via VNC.
  - Use the Perl script created by Ben Low to decode: **cat com.apple.VNCSettings.txt | perl -wne 'BEGIN { @k = unpack "C*", pack "H*", "1734516E8BA8C5E2FF1C39567390ADCA"}; chomp; @p = unpack "C*", pack "H*", $_; foreach (@k) { printf "%c", $_ ^ (shift @p || 0) }; print "\n"'**

Screen sharing events:
- Unified Logs
  - Search for "screensharingd"

Interpretation
disabled.plist:
- By default, none of these settings are enabled.
- **com.apple.screensharing** = NO (0) – Screen sharing is enabled.
- **com.openssh.sshd** = NO (0) – Remote Login is enabled.
- If the bundle ID for a service does not appear in the list, it was likely never enabled.

Keychains

Description
The keychains on a system are used to store sensitive data such as usernames, passwords, and encryption keys.

Location
macOS:
- **~/Library/Keychains/login.keychain-db**
- iCloud: **~/Library/Keychains/<Hardware UUID>/keychain-2.db**
- **/Library/Keychains/System.keychain**

iOS FFS:
- **/private/var/Keychains/keychain-2.db**

iOS encrypted backup:
- **Keychain*/keychain-backup.plist**

Interpretation
- **login.keychain-db** may contain user passwords for access points, Time Machine, applications, and websites.
- Default **login.keychain-db** password is the user's account password.
- **System.keychain** contains passwords for VPNs, access points, Time Machine, and applications.
- **iCloud keychain-2.db** may contain information from other iDevices.
- On iOS backups, the keychain may be stored in a Keychains or KeychainDomain folder, depending on the acquisition tool used.
  - iCloud keychain items are not specifically backed up in this plist file.
- View a keychain file using the **Keychain Access.app** macOS application or using the strings or security commands if the keychain is not encrypted.

Accounts Configured on the System

Description
A user can configure accounts on the system, such as email, calendar, and iCloud.

Location
macOS:
- **~/Library/Accounts/Accounts4.sqlite**
- **/Library/Preferences/SystemConfiguration/com.apple.accounts.exists.plist**

iOS:
- **/private/var/mobile/Library/Accounts/Accounts3.sqlite**
- **/private/var/Preferences/SystemConfiguration/com.apple.accounts.exists.plist**

Interpretation
- ZACCOUNT table in the sqlite databases contains account information.
  - ZUSERNAME is the account username.
  - ZACCOUNTTYPEDESCRIPTION is the account type description.
  - ZDATE is the account setup date in Mac Epoch format.
  - ZKEY is the configuration key name.
  - ZVALUE is the configuration value, as a BLOB that contains a binary plist.
- **com.apple.accounts.exists.plist** file has two associated keys per account:
  - Exists shows if the account is in use.
  - Count shows how many of this account type there are.

[iOS] Apple Watch Data

Description
If an Apple Watch is paired with an iPhone (it cannot be paired with any other iOS devices), some data will be synced with that iPhone.

Location
- **/private/var/mobile/Library/DeviceRegistry/<GUID>/***
- **/private/var/mobile/Library/DeviceRegistry.state/<GUID>/***

Interpretation
- **historySecureProperties.plist** includes device serial number, IMEI, Bluetooth MAC address, and WiFi MAC address of the device.

†**https://github.com/mac4n6/APOLLO**

Poster was created by Kathryn Hedley and Sarah Edwards based on many years of research and macOS and iOS knowledge by Sarah Edwards.