518.2

System Triage and File Systems

SANS | GIAC CERTIFICATIONS

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

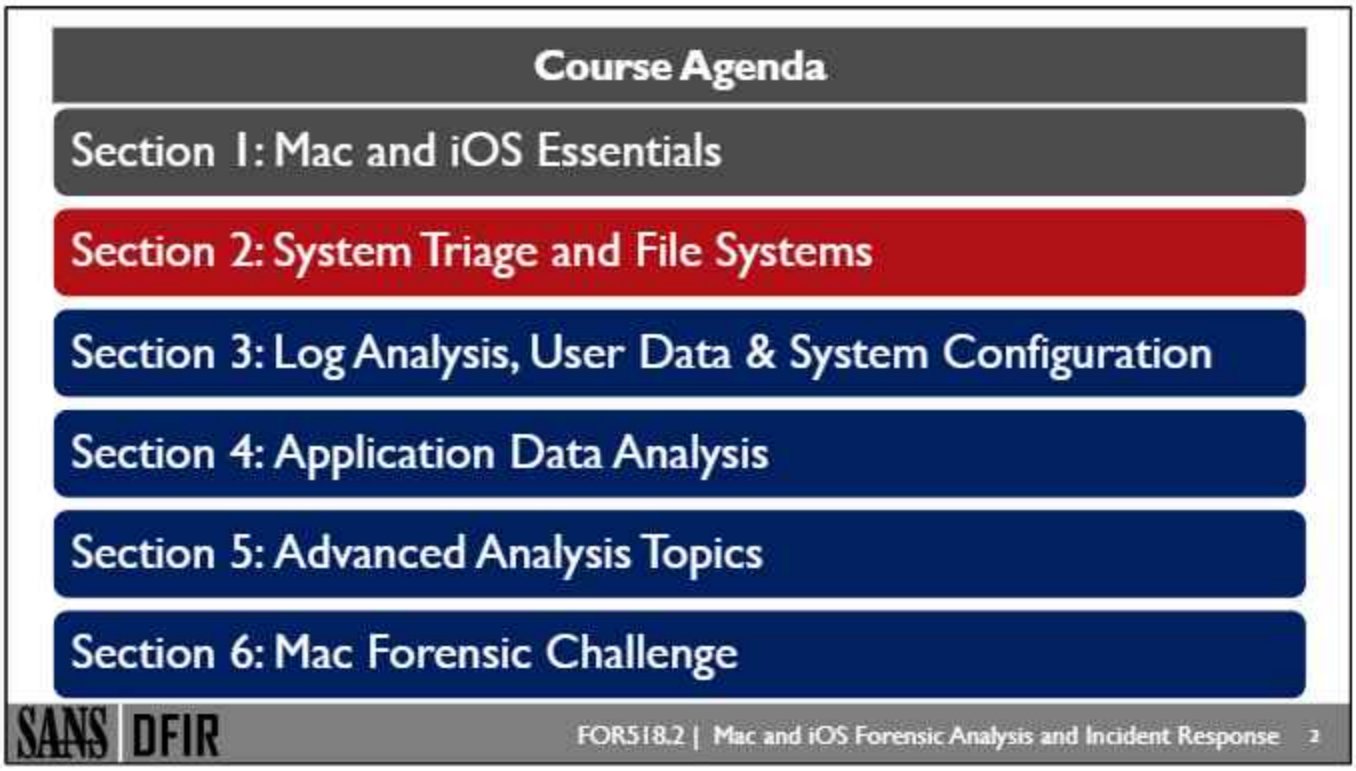
All reference links are operational in the browser-based delivery of the electronic workbook.

FOR518.2 | Mac and iOS Forensic Analysis and Incident Response

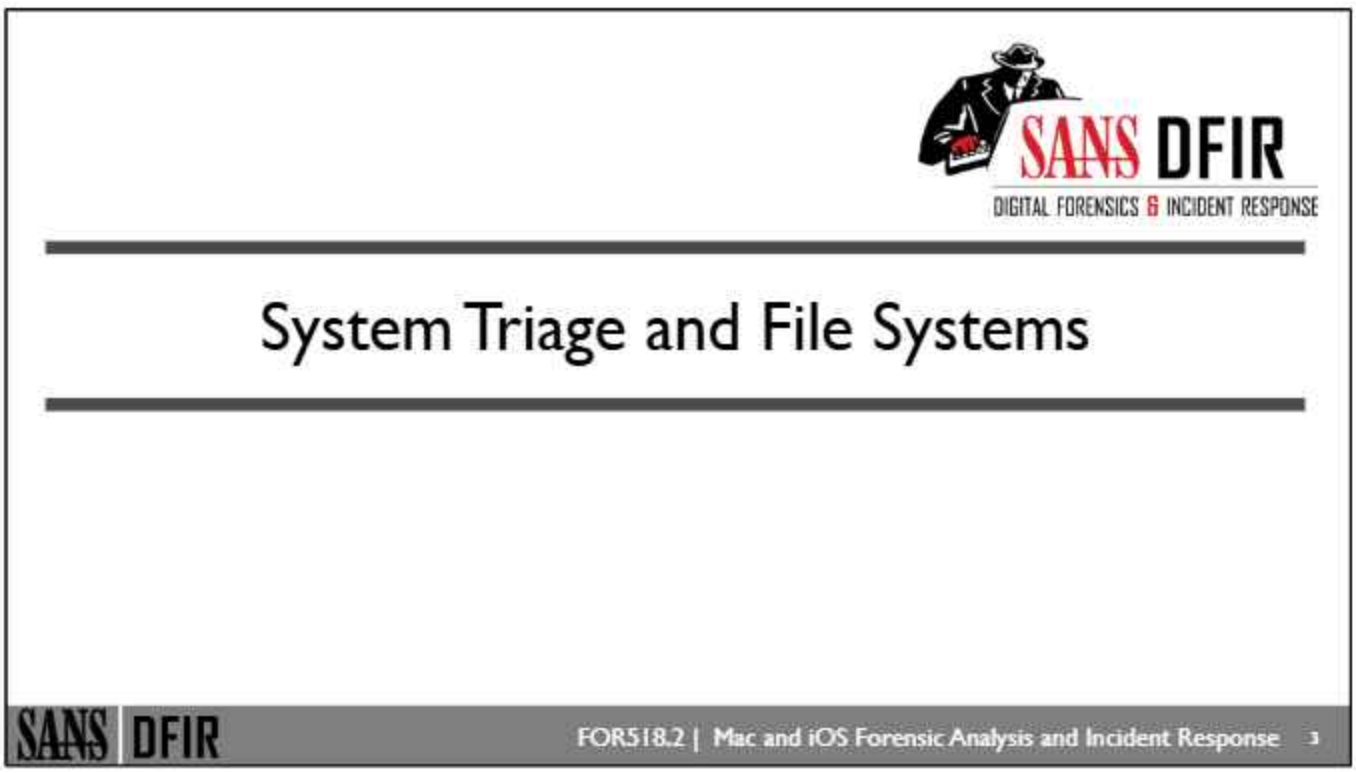
System Triage and File Systems

Author: Sarah Edwards
sedwards@sans.edu
mac4n6.com
https://twitter.com/iamevltwin

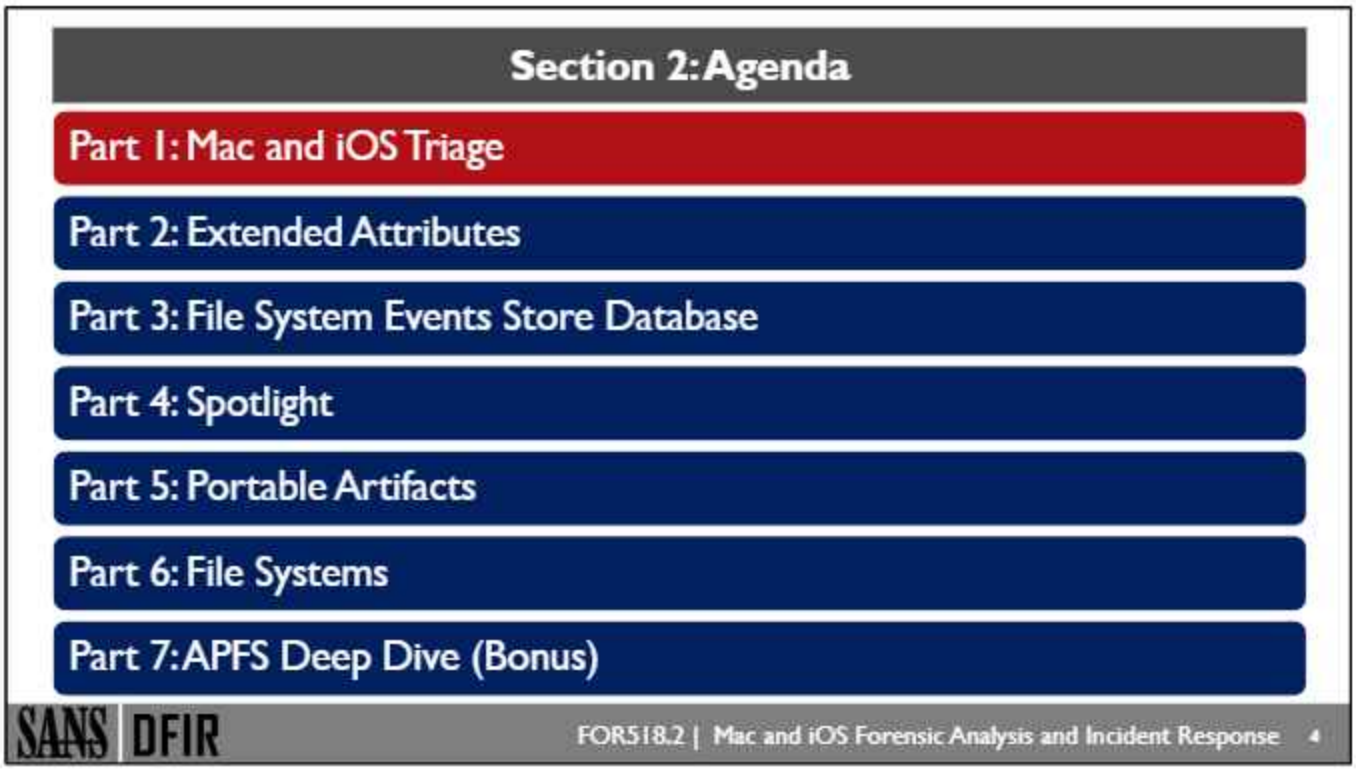
https://digital-forensics.sans.org/
https://twitter.com/sansforensics



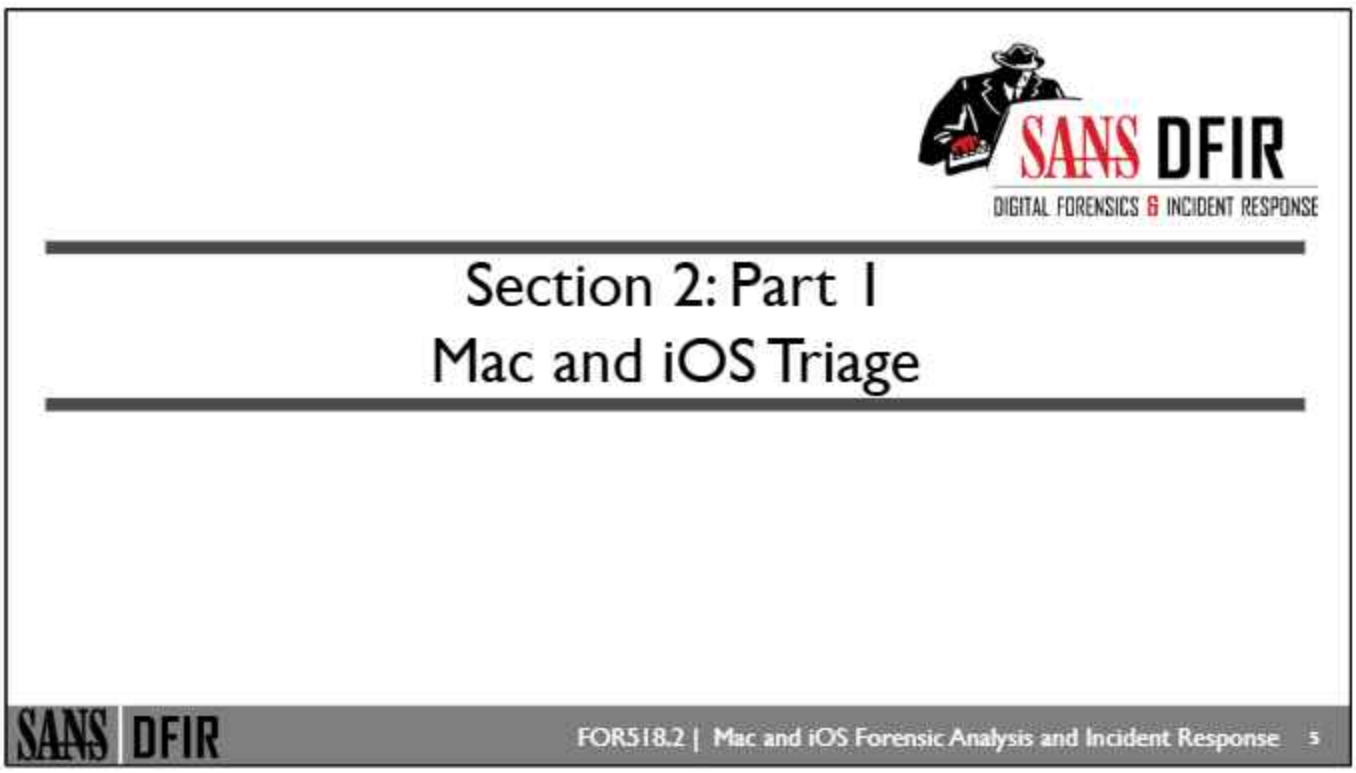
This page intentionally left blank.



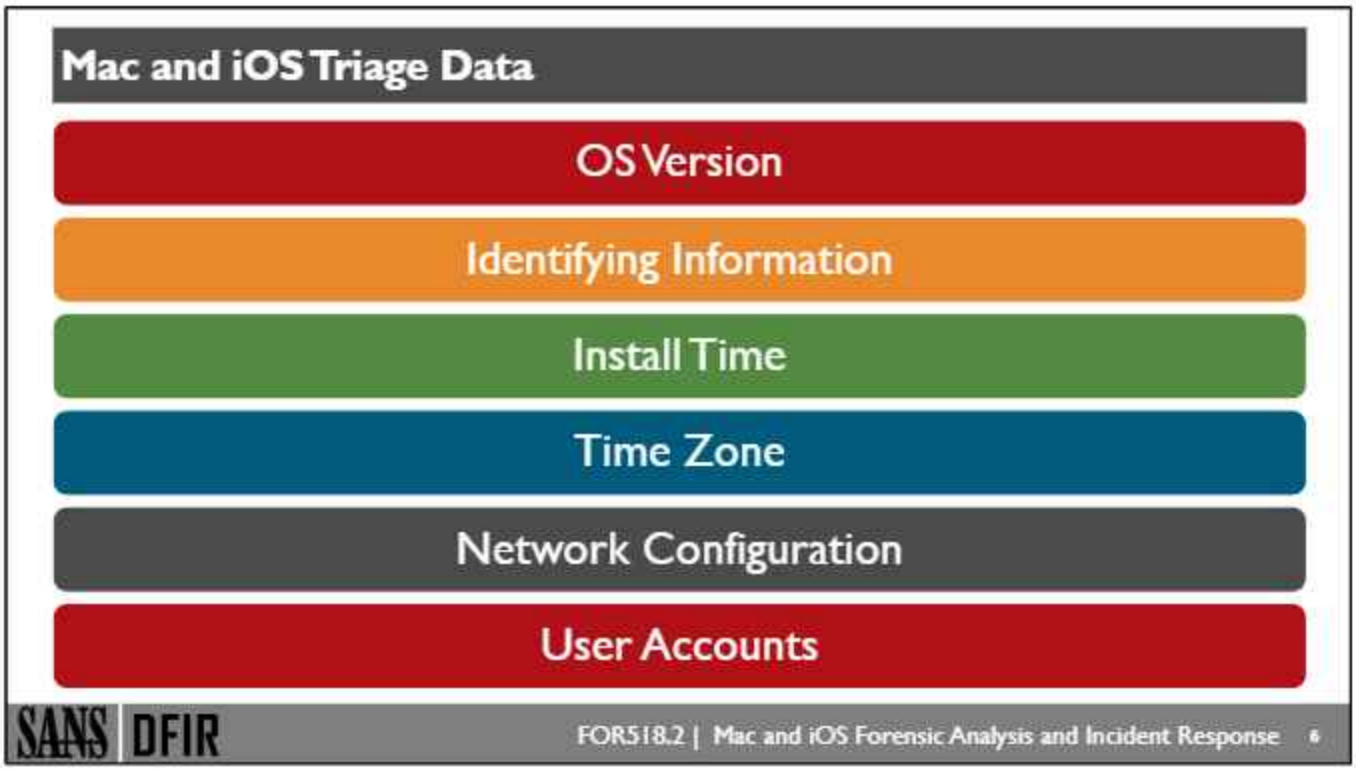
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.



Extracting triage data from Mac and iOS can be both very similar and completely different. In this module, we will extract the basics of each system to get an idea of what we are looking at to include OS versions, configurations, and identifying information.

MacOS and iOS Information – Version and Serial Number /System/Library/CoreServices/SystemVersion.plist

Root	Dictionary	(6 items)
ProductBuildVersion	String	21A559
ProductCopyright	String	1983-2021 Apple Inc.
ProductName	String	macOS
ProductUserVisibleVersion	String	12.0.1
ProductVersion	String	12.0.1
iOSSupportVersion	String	15.0

Device Serial Number

- Location Database - cache_encryptedA.db and others!

Database Structure | Browse Data | Edit P

Table: TableInfo | New

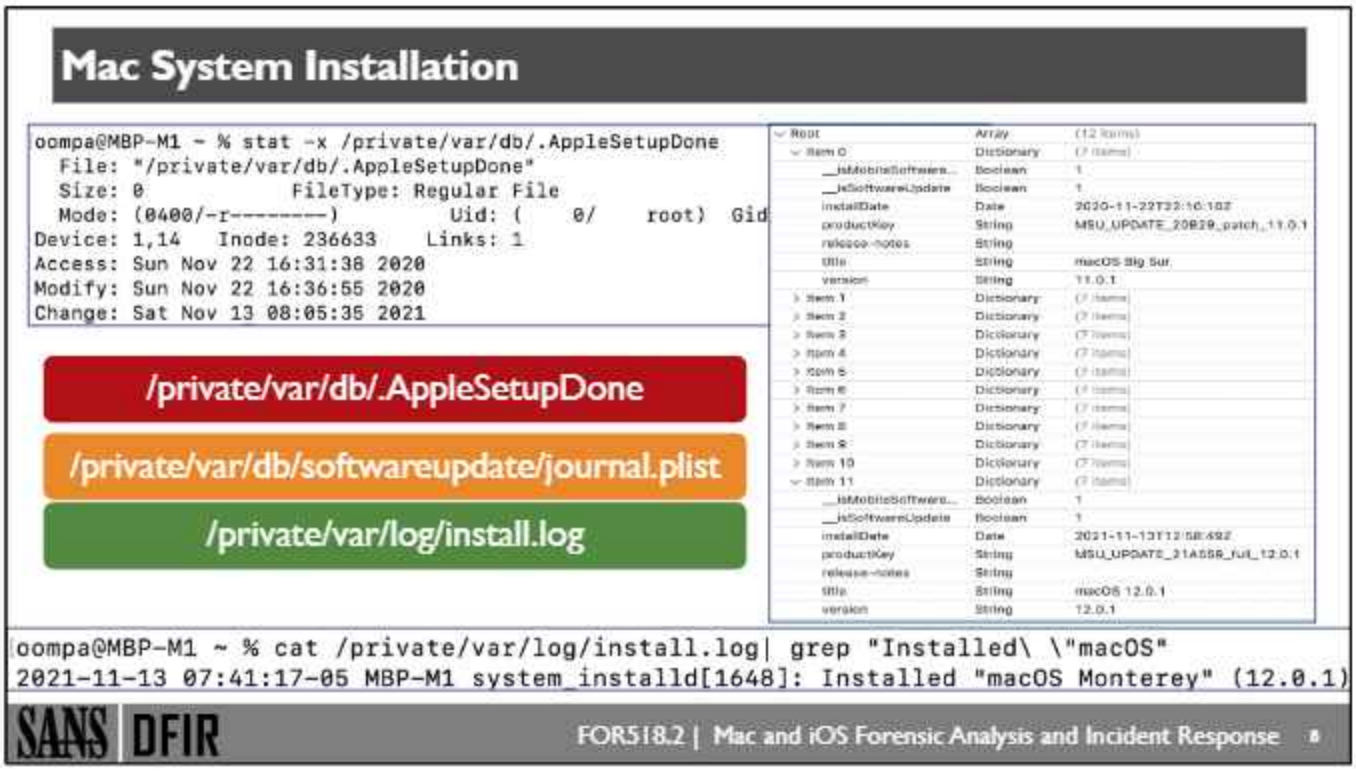
	TableName	SoftwareVersion	SerialNumber
	Filter	Filter	Filter
1	WifiLocation	2236.0.11	C02SR5VYHF1R

SANS DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 7

The `SystemVersion.plist` property list located in the `/System/Library/CoreServices/` directory contains the system version and build information.

In the example above, the system name and version is macOS 12.0.1, while the build version is 21A559.

The Mac's serial number can be found by extracting it from a live system, perhaps from the system_profiler output. On newer versions of macOS, it can be found in a variety of different databases. The example shown above is from the cache_encryptedA.db location database. It will usually be found in a small table named TableInfo.

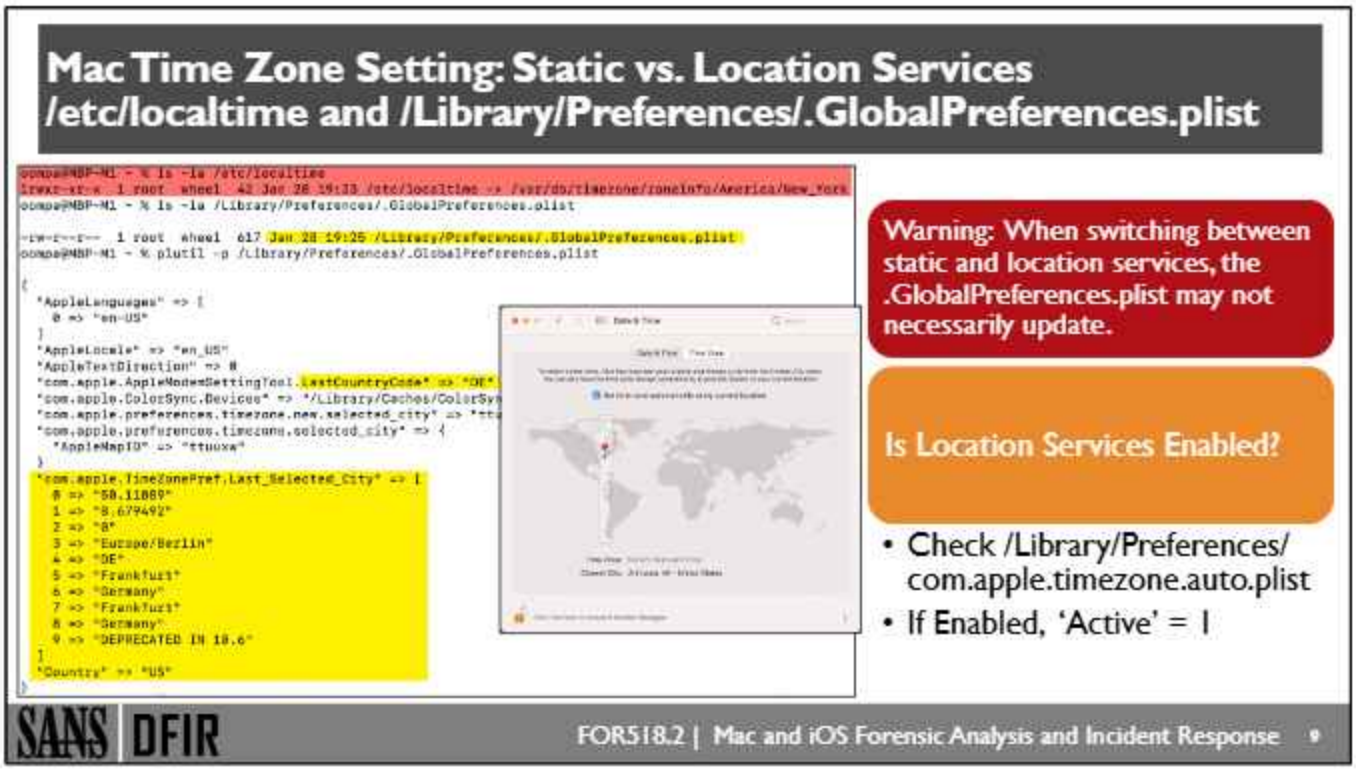


An original system setup/registration date may be found as the access/modification date for the `.AppleSetupDone` file located in the `/private/var/db/` directory. The change time is likely the last installation of a system update.

The macOS installation date may be able to be found in the `install.log` files located in the `/var/log` directory if the older files have not been turned over. These dates are when different versions of macOS were installed after the original installation.

There may be a difference in timestamp time zones, as the original time zone for all Macs is Cupertino, CA time, before the user sets up their own time zone.

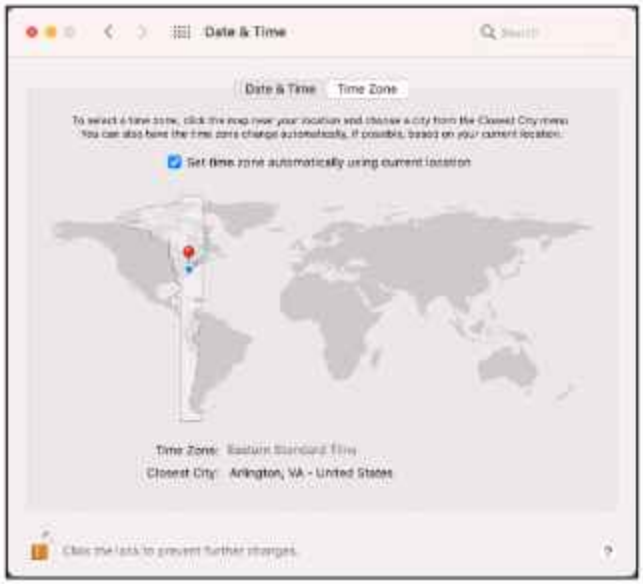
Another file worth looking at is `/private/var/db/softwareupdate/journal.plist` – this may show you detailed installation timestamps for system software.

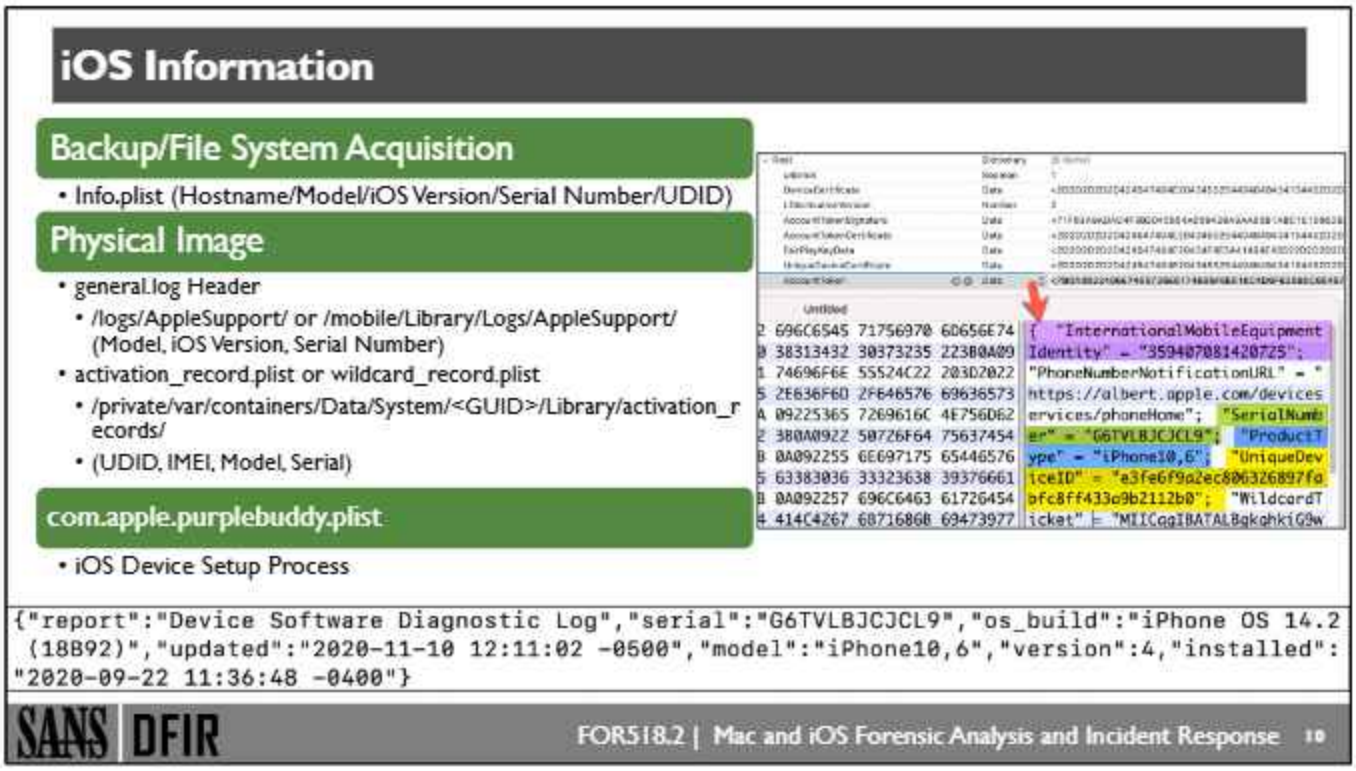


The link to the file located in `/etc/localtime` contains the current time zone value for the system. In the screenshot, you can see the local time zone is set for New York, or the Eastern Time zone. The `.GlobalPreferences` property list located in the `/Library/Preferences/` directory contains the time zone configuration data. The time zone is set for Frankfurt, Germany. This is a good example of this plist not updating after switching from using location services (automatic time zone updates) to a static location.

The `com.apple.timezone.auto.plist` can be used to determine if location services are being used.

If the user has location settings turned on, it will use Wi-Fi access point beacons to determine where the user is. The coordinates in the plist are usually a city center if the user selects a specific city (in this example, Frankfurt) or somewhat closer if the user is using location services.





Simple device information such as model, iOS version, device serial number, UDID, and hostname can be found in a variety of locations, depending on whether you have a file system/backup acquisition or a full physical image.

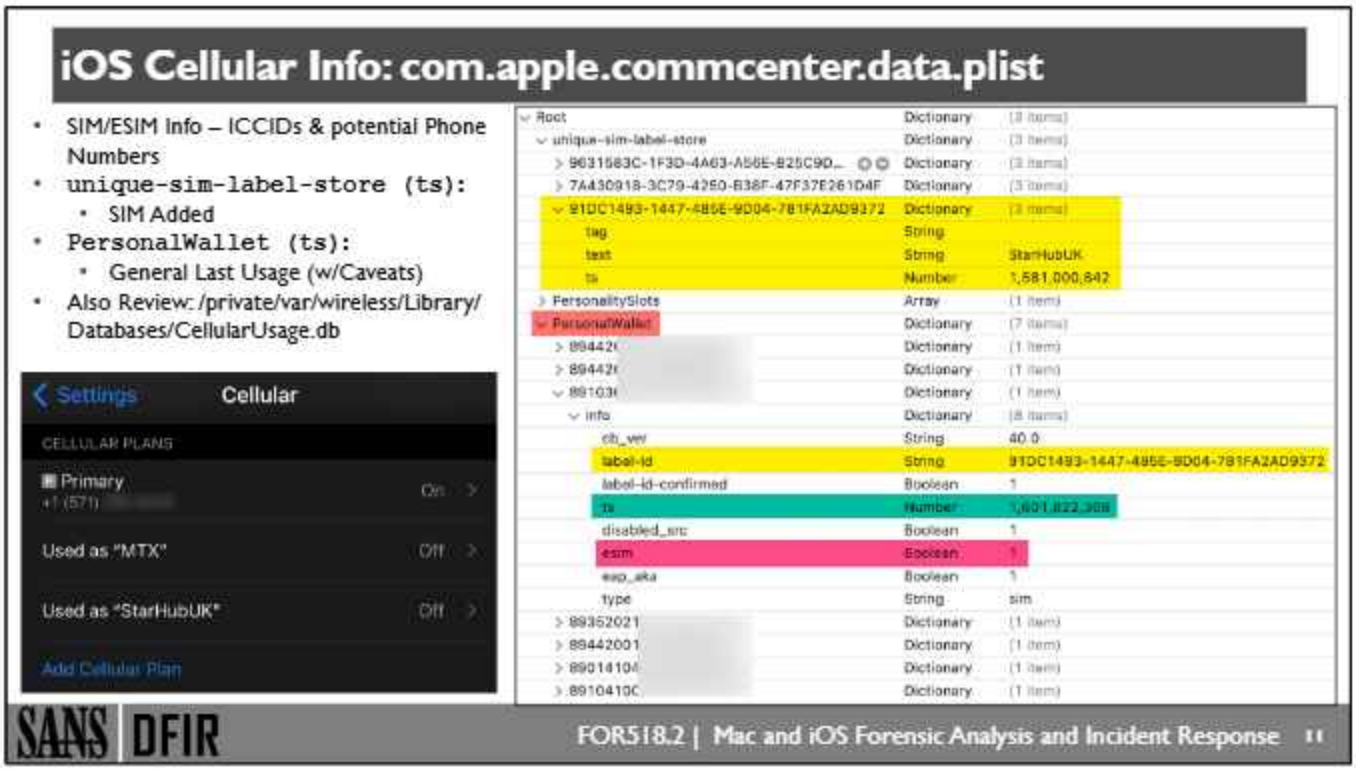
On backups or file system acquisitions, this information can easily be found in the `Info.plist` that is created. Some acquisition tools do not necessarily call this the `Info.plist`, but they should have a similar file containing the identifying information for the device.

On physical images, this information is stored in many files:

- The model, iOS version, and serial number can be found in the `general.log` file located in `/logs/AppleSupport/` or `/mobile/Library/Logs/AppleSupport/` as the header of the log file.
- The UDID can be found in the `activation_record.plist` or `wildcard_record.plist` file located in the /private/var/containers/Data/System/<GUID>/Library/activation_records/ directory as embedded data in the `AccountToken` key.

`/[private/var]/mobile/Library/Preferences/com.apple.purplebuddy.plist`

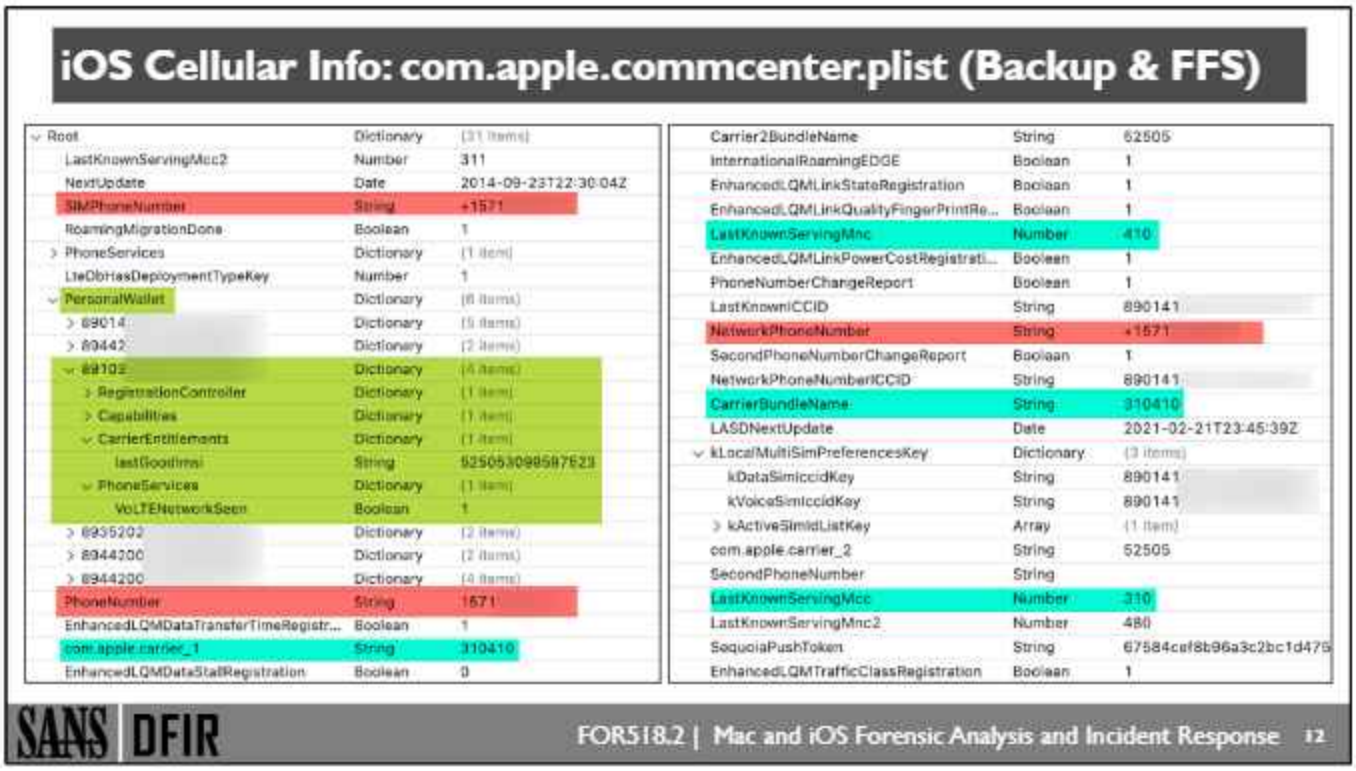
The `com.apple.purplebuddy.plist` can be viewed to determine the device setup information, including the original locale, setup time, and device hardware model. This purplebuddy setup process is where the user configures all items about their devices, including their passcode, locale, keychain, Wi-Fi, etc.



Physical:
`/private/var/wireless/Library/Preferences/com.apple.commcenter.data.plist`
Backup: `/wireless/Library/Preferences/com.apple.commcenter..data.plist`

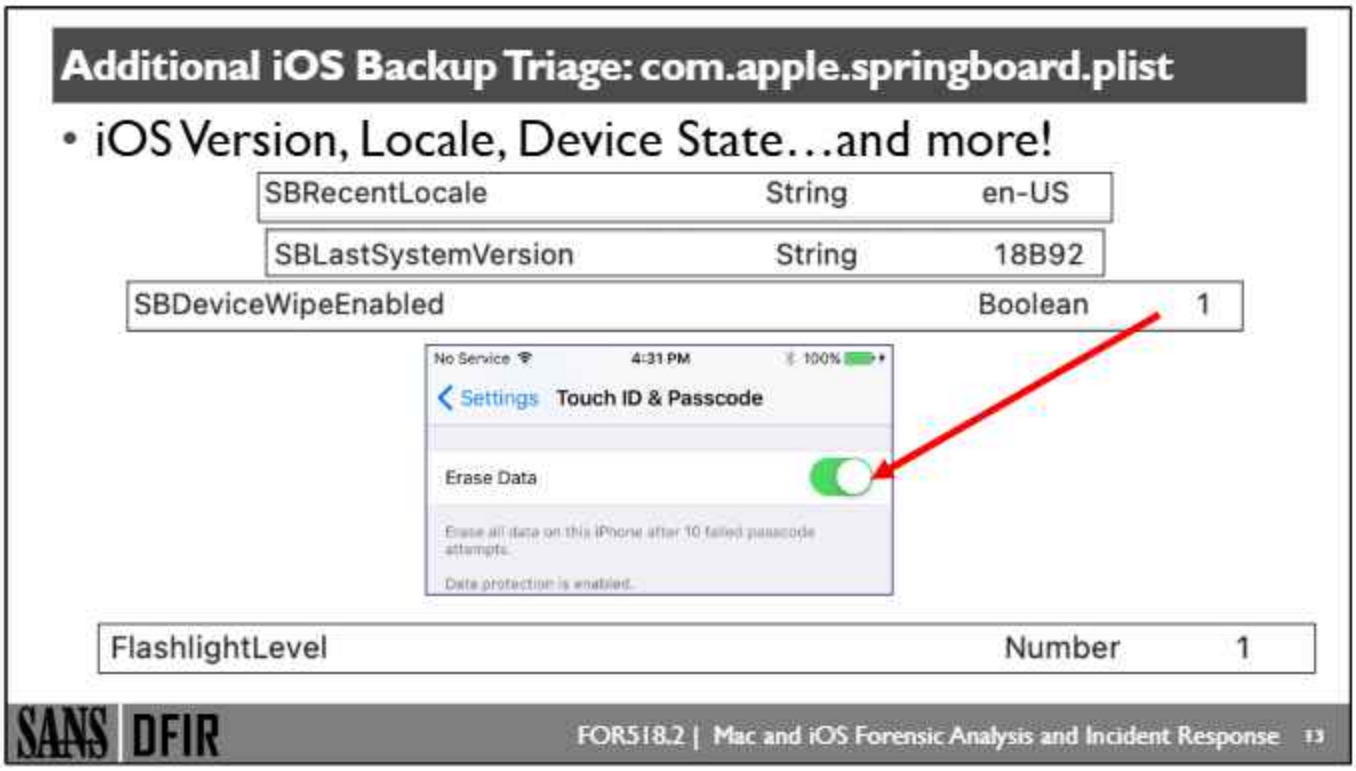
The `com.apple.commcenter.data.plist` can be viewed to determine the current and historical ICCID, Phone Numbers, and carrier information.

Be careful of the timestamps; some timestamps may not necessarily reflect expected SIM usage.



Physical: `/private/var/wireless/Library/Preferences/com.apple.commcenter.plist`
Backup: `/wireless/Library/Preferences/com.apple.commcenter.plist`

The `com.apple.commcenter.plist` can be viewed to determine the phone number, carrier, IMSI, and ICCID information. This plist file may also contain carrier information. The "CarrierBundleName" can be used to search for the carrier on https://www.mcc-mnc.com/. For example, 310410 belongs to the AT&T network.



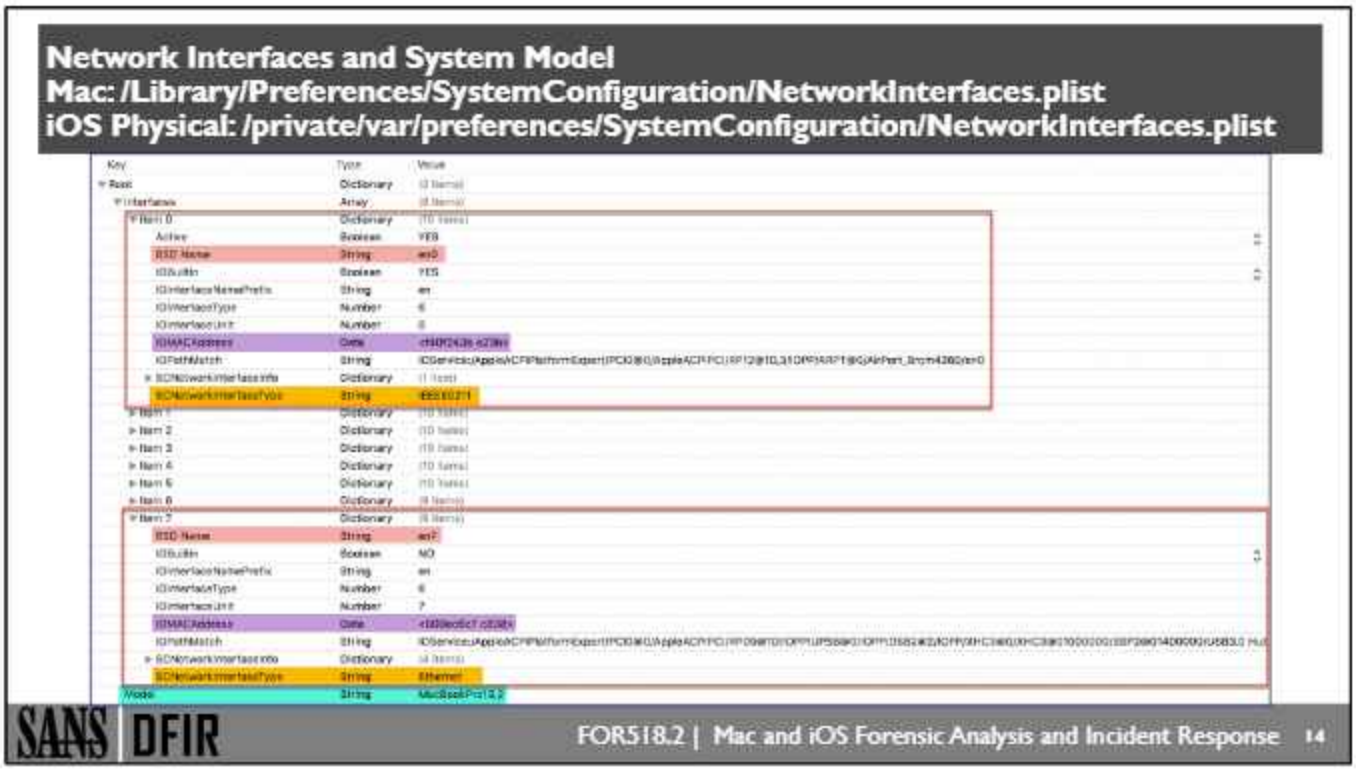
Location: `[/private/var]/mobile/Library/Preferences/com.apple.springboard.plist`

Various Springboard and interface-related items are stored in this plist file.

Additional iOS triage information can be found in the `com.apple.springboard.plist` file. The locale setting (`SBRecentLocale`) can provide native location-based information. In this example, it is "`en-US`". The OS version (`SBLastSystemVersion`) is stored as a build number—"`18B92`". This number can be Google searched to determine the numerical OS version (14.2).

The `SBDeviceWipeEnabled` key determines if the user set the "Erase Data" option in their "Touch ID and Passcode" settings to wipe the device after 10 failed attempts at the passcode.

And just as an added bonus, it does keep some seemingly random settings as well, like what level the Flashlight is set to!



Mac: `/Library/Preferences/SystemConfiguration/NetworkInterfaces.plist`
iOS Physical: `/private/var/preferences/SystemConfiguration/NetworkInterfaces.plist`

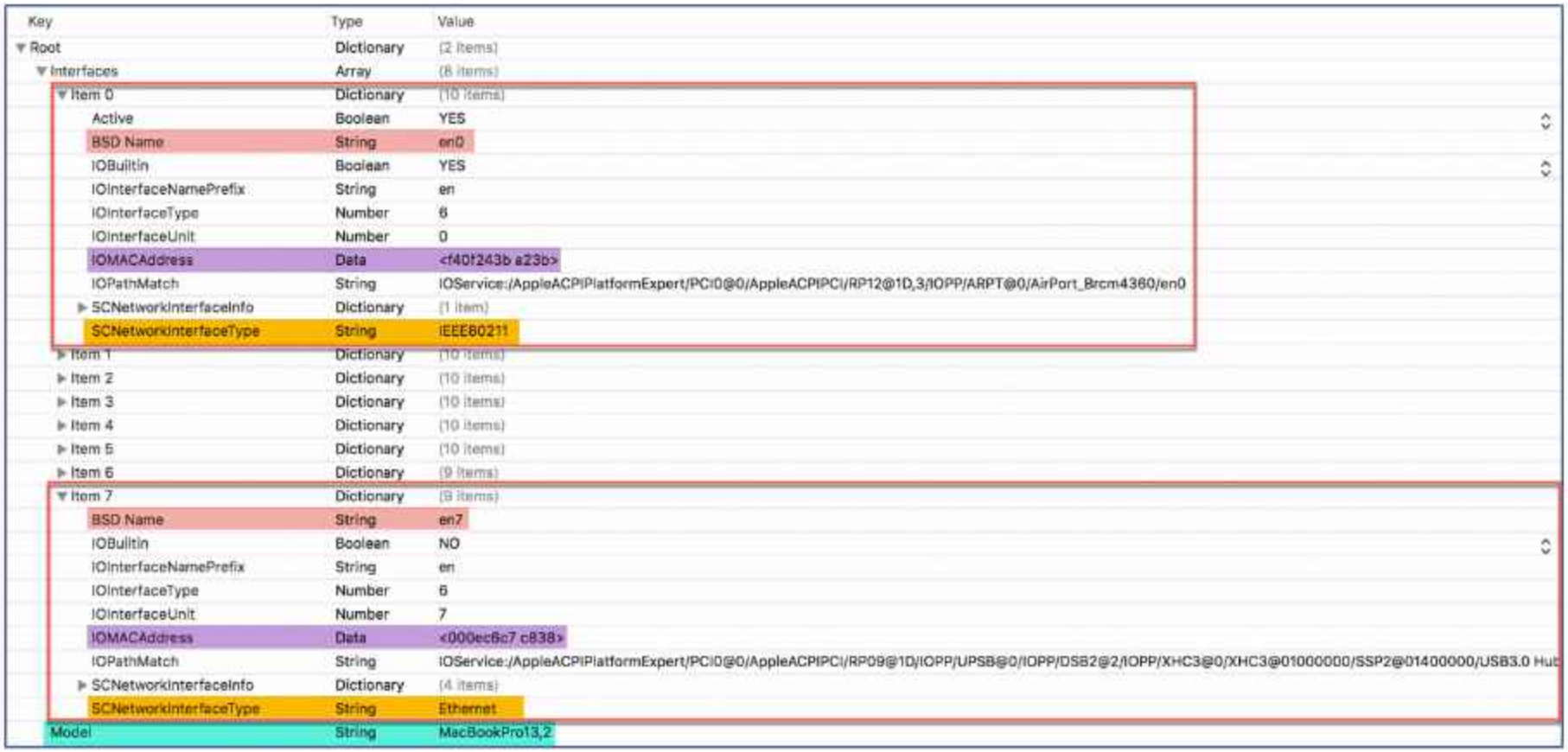
The `NetworkInterfaces.plist` file located in the `SystemConfiguration` directory contains the network interfaces for the system.

Each network interface on the system will have an `Item` key. There are two interfaces highlighted in the above screenshot.

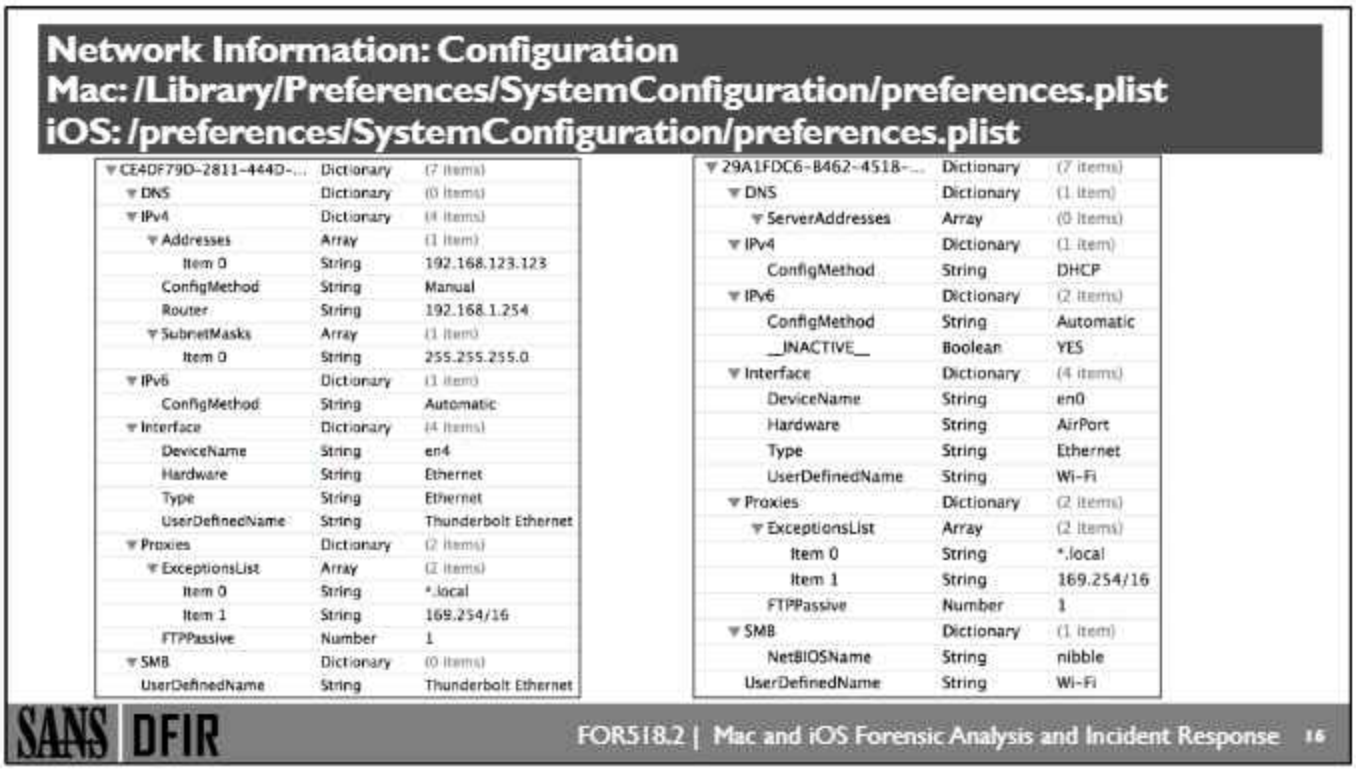
"`Item 0`" is the Wi-Fi interface on en0, while "`Item 7`" is a USB-C hub that contains a network port that was attached to the system.

Each interface will contain a description such as "`IEEE802.11`" or "`Ethernet`", as well as the unique MAC address for each interface.

Finally, there is a Model key that shows the model of the system (i.e., MacBookPro13,2). This can be searched for on "https://support.apple.com/en-us/HT201300" to determine the actual model information.



Key	Type	Value
▼ Root	Dictionary	(2 items)
▼ Interfaces	Array	(8 items)
▼ Item 0	Dictionary	(10 items)
Active	Boolean	YES
BSD Name	String	en0
IOBuiltin	Boolean	YES
IOInterfaceNamePrefix	String	en
IOInterfaceType	Number	6
IOInterfaceUnit	Number	0
IOMACAddress	Data	<f40f243b a23b>
IOPathMatch	String	IOService:/AppleACPIPlatformExpert/PCI0@0/AppleACPIPCI/RP12@1D,3/IOPP/ARPT@0/AirPort_Brcm4360/en0
▶ SCNetworkInterfaceInfo	Dictionary	(1 item)
SCNetworkInterfaceType	String	IEEE80211
▶ Item 1	Dictionary	(10 items)
▶ Item 2	Dictionary	(10 items)
▶ Item 3	Dictionary	(10 items)
▶ Item 4	Dictionary	(10 items)
▶ Item 5	Dictionary	(10 items)
▶ Item 6	Dictionary	(9 items)
▼ Item 7	Dictionary	(9 items)
BSD Name	String	en7
IOBuiltin	Boolean	NO
IOInterfaceNamePrefix	String	en
IOInterfaceType	Number	6
IOInterfaceUnit	Number	7
IOMACAddress	Data	<000ec6c7 c838>
IOPathMatch	String	IOService:/AppleACPIPlatformExpert/PCI0@0/AppleACPIPCI/RP09@1D/IOPP/UPSB@0/IOPP/DSB2@2/IOPP/XHC3@0/XHC3@01000000/SSP2@01400000/USB3.0 Hub
▶ SCNetworkInterfaceInfo	Dictionary	(4 items)
SCNetworkInterfaceType	String	Ethernet
Model	String	MacBookPro13,2



Mac: `/Library/Preferences/SystemConfiguration/preferences.plist`
iOS Backup: `/preferences/SystemConfiguration/preferences.plist`
iOS Physical: `/private/var/preferences/SystemConfiguration/preferences.plist`

The `NetworkServices` key contains the configuration for each network interface. Two examples are shown above.

The example on the left contains a Wi-Fi interface (`UserDefinedName` key). This interface uses DHCP (rather than a static IP), and it has a `NetBIOSName` key that contains the NetBIOS name of the system. The network interface device is `en0`.

The example on the right contains a Thunderbolt Ethernet interface (`UserDefinedName` key). This interface has a static IP configured (as well as router and subnet mask). The network interface device is `en4`.

Network Information: DHCP Addresses
Mac: /private/var/db/dhcpclient/leases/
iOS Physical: /db/dhcpclient/leases/

```
root@MBP-M1 leases # pwd
/private/var/db/dhcpclient/leases
root@MBP-M1 leases # plutil -p en0.plist
{
  "ClientIdentifier" => {length = 7, bytes = 0x01c4910cb38811}
  "IPAddress" => "192.168.1.227"
  "LeaseLength" => 86400
  "LeaseStartDate" => 2021-03-06 14:32:59 +0000
  "PacketData" => {length = 312, bytes = 0x02010600 368fdf4f 00000000 c0a801e3 ... 7204686f 6d6500ff }
  "RouterHardwareAddress" => {length = 6, bytes = 0xc8a70acb23e4}
  "RouterIPAddress" => "192.168.1.1"
  "SSID" => "Fios-4JYWD-5G"
}
```


Mac: `/private/var/db/dhcpclient/leases/`
iOS Physical: `/db/dhcpclient/leases/`

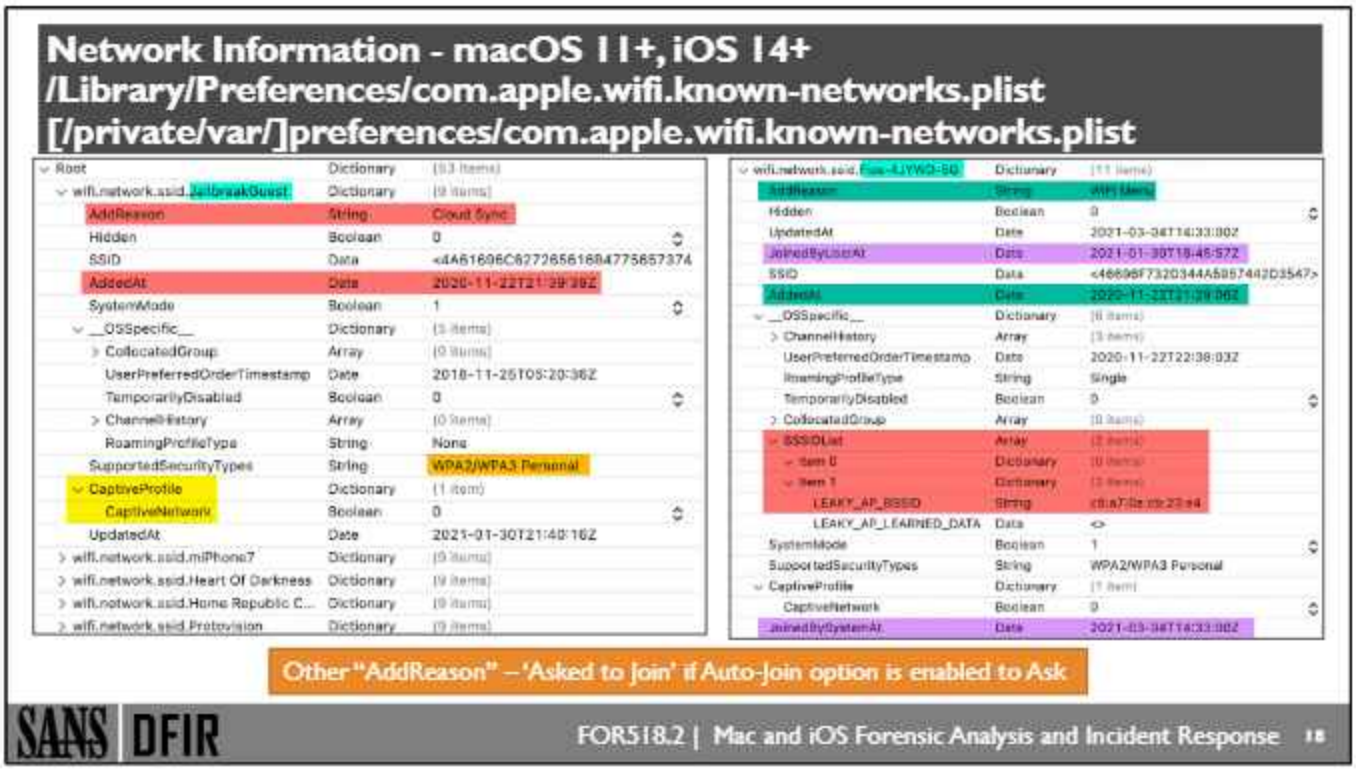
The files located in the directories above contain configurations for DHCP network settings. These contain the settings for the latest connection on the specified interface.

Each file contains:

- Lease Start Date, Router MAC Address, Assigned IP Address, SSID of Access Point, DHCP Lease Length,Router IP Address, Packet Data

The filename of these tiny files changed in macOS 11; previously, they used to contain the MAC address for the interface.

```
bash-3.2# pwd
/private/var/db/dhcpclient/leases
bash-3.2# ls -l
total 16
-rw-r--r--  1 root  wheel  969 May 10 10:20 en0-1,b8:e8:56:37:ec:6
-rw-r--r--  1 root  wheel  927 Feb 18 20:48 en4-1,68:5b:35:91:1a:b5
bash-3.2# plutil -p en4-1\,68\:5b\:35\:91\:1a\:b5
{
  "LeaseStartDate" => 2014-02-19 01:39:52 +0000
  "RouterHardwareAddress" => <e0699550 4c06>
  "IPAddress" => "192.168.1.237"
  "LeaseLength" => 43200
  "RouterIPAddress" => "192.168.1.254"
  "PacketData" => <02010600 7a48b9f4 000d0000 00000000 c0a801ed c0a801fe 00000000 685b3591 1ab
50000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000
0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00
000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000
00 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0
0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 63825363 35010
536 04c0a801 fe330400 00a8c03a 04000054 603b0400 0093a801 04ffffff 001c04c0 a801ff03 04c0a801
fe0604c0 a801feff 00000000 00000000>
}
```

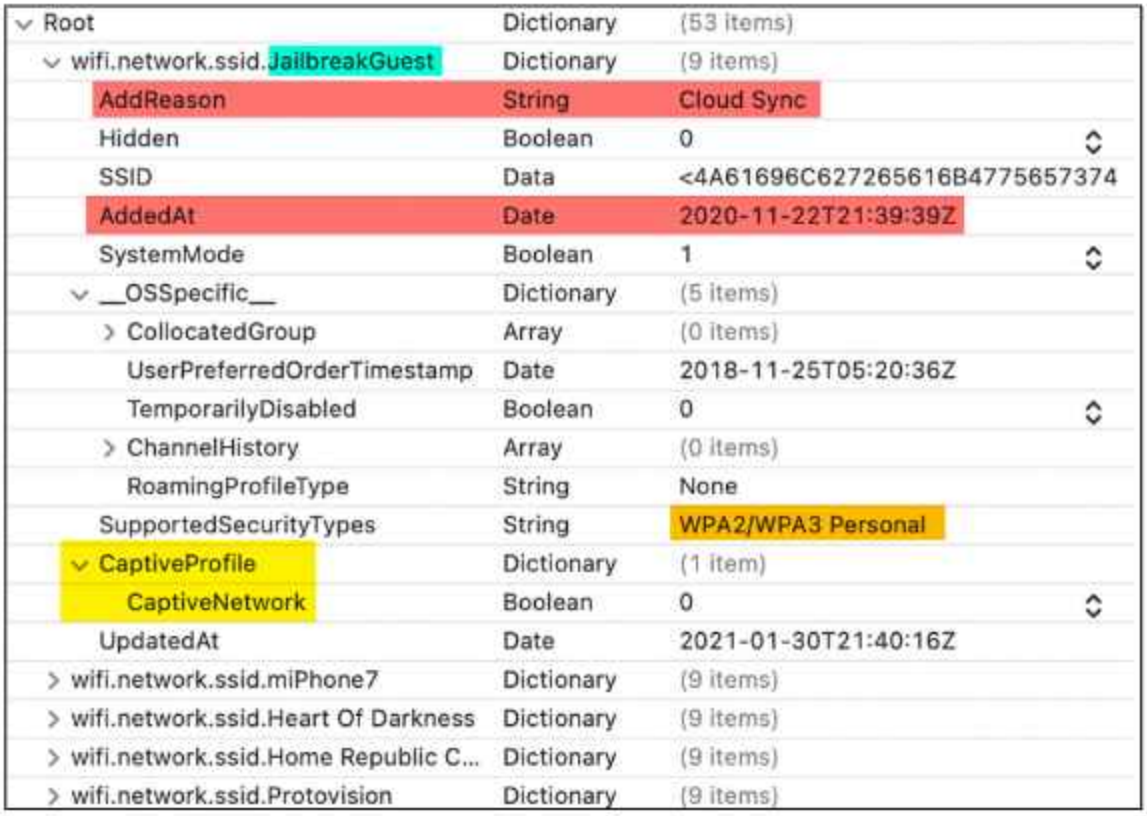


Connections to various access points are stored in the `com.apple.wifi.known-networks.plist` file. Each access point's information is stored in its own key. This information can include expected Wi-Fi information such as security, captive network settings, BSSID's, etc.

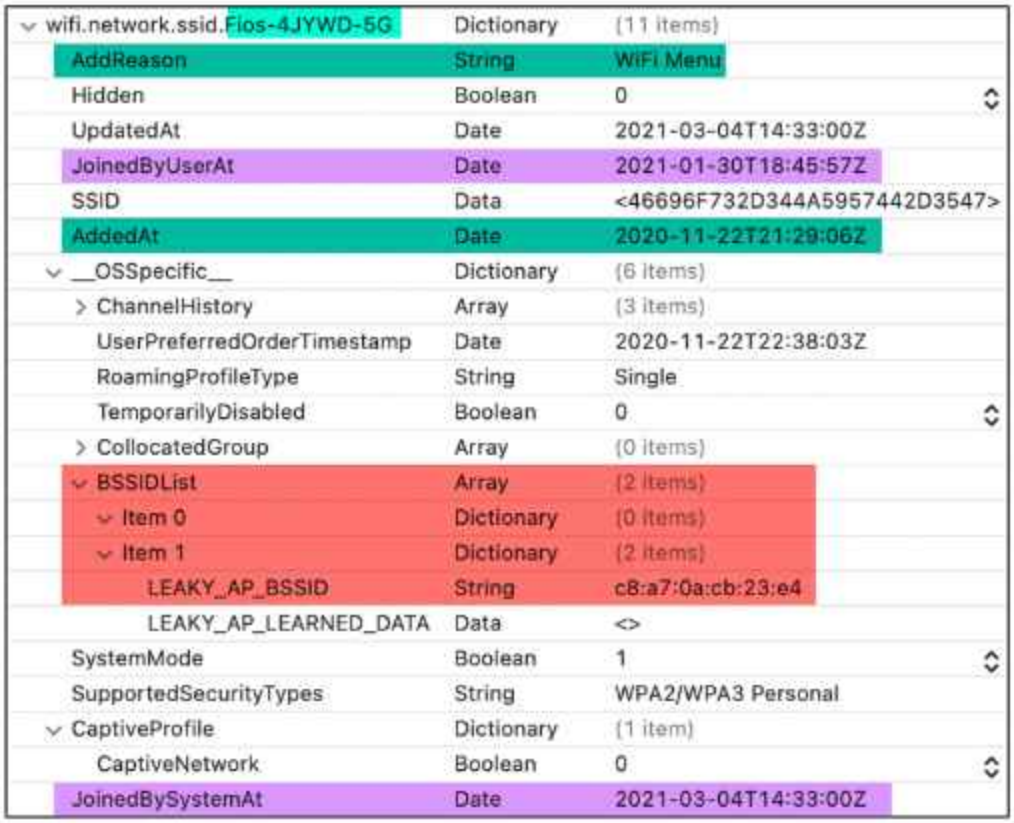
A few additional keys are worth pointing out. Note the `AddReason` key. On the left is an AP that has been synced via iCloud and on the right is one that was added via the Wi-Fi menu. These also include an `AddedAt` timestamp.

An access point listed in this plist file has not necessarily been connected via the device that it was extracted from. Look for keys that have the word 'Joined' in them to determine how and when they were connected.

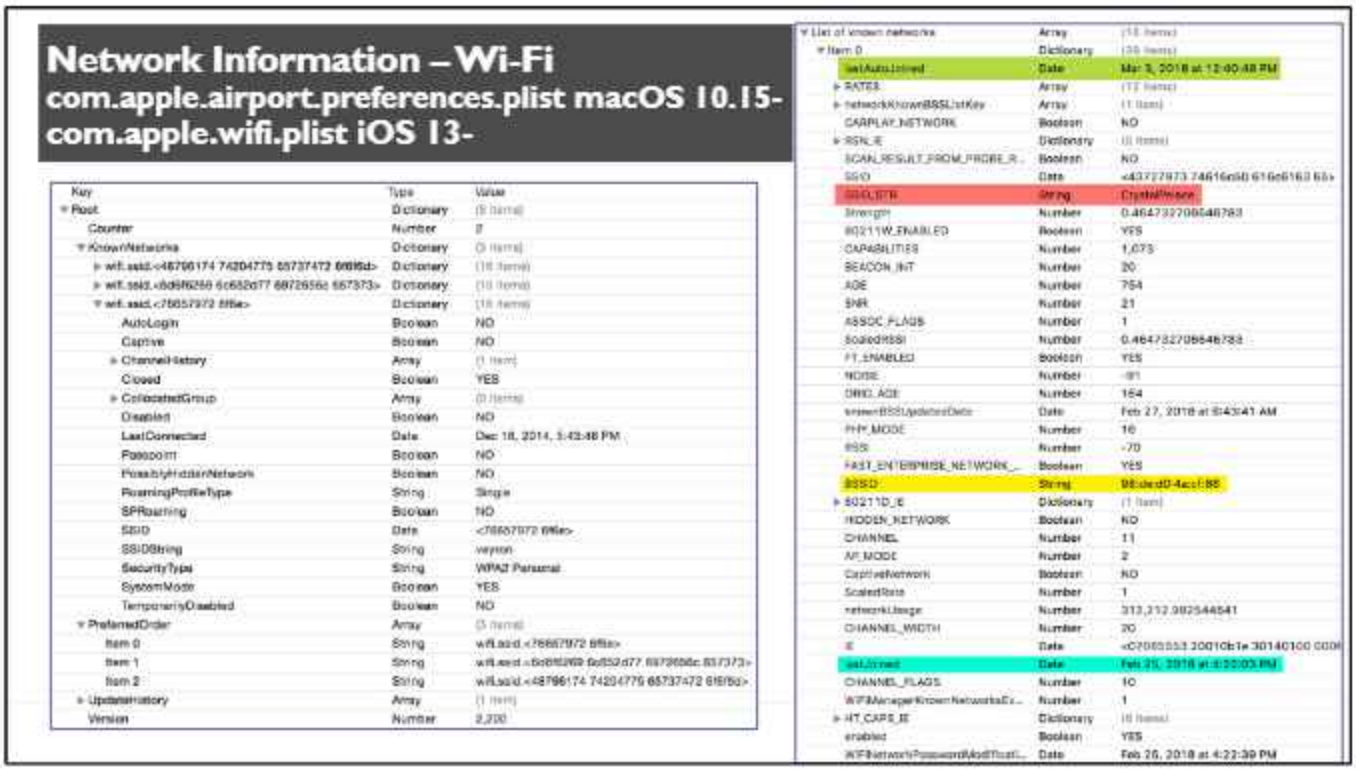
- `JoinedByUserAt` – The user specifically chose this AP to join.
- `JoinedBySystemAt` – The system auto-connected to this AP.



Root	Dictionary	(53 items)
wifi.network.ssid.JailbreakGuest	Dictionary	(9 items)
AddReason	String	Cloud Sync
Hidden	Boolean	0
SSID	Data	<4A61696C62726561B4775657374
AddedAt	Date	2020-11-22T21:39:39Z
SystemMode	Boolean	1
__OSSpecific__	Dictionary	(5 items)
CollocatedGroup	Array	(0 items)
UserPreferredOrderTimestamp	Date	2018-11-25T05:20:36Z
TemporarilyDisabled	Boolean	0
ChannelHistory	Array	(0 items)
RoamingProfileType	String	None
SupportedSecurityTypes	String	WPA2/WPA3 Personal
CaptiveProfile	Dictionary	(1 item)
CaptiveNetwork	Boolean	0
UpdatedAt	Date	2021-01-30T21:40:16Z
wifi.network.ssid.miPhone7	Dictionary	(9 items)
wifi.network.ssid.Heart Of Darkness	Dictionary	(9 items)
wifi.network.ssid.Home Republic C...	Dictionary	(9 items)
wifi.network.ssid.Protovision	Dictionary	(9 items)



wifi.network.ssid.Fios-4JYWD-5G	Dictionary	(11 items)
AddReason	String	WiFi Menu
Hidden	Boolean	0
UpdatedAt	Date	2021-03-04T14:33:00Z
JoinedByUserAt	Date	2021-01-30T18:45:57Z
SSID	Data	<46696F732D344A5957442D3547>
AddedAt	Date	2020-11-22T21:29:06Z
__OSSpecific__	Dictionary	(6 items)
ChannelHistory	Array	(3 items)
UserPreferredOrderTimestamp	Date	2020-11-22T22:38:03Z
RoamingProfileType	String	Single
TemporarilyDisabled	Boolean	0
CollocatedGroup	Array	(0 items)
BSSIDList	Array	(2 items)
Item 0	Dictionary	(0 items)
Item 1	Dictionary	(2 items)
LEAKY_AP_BSSID	String	c8:a7:0a:cb:23:e4
LEAKY_AP_LEARNED_DATA	Data	<>
SystemMode	Boolean	1
SupportedSecurityTypes	String	WPA2/WPA3 Personal
CaptiveProfile	Dictionary	(1 item)
CaptiveNetwork	Boolean	0
JoinedBySystemAt	Date	2021-03-04T14:33:00Z



Mac:
`/Library/Preferences/SystemConfiguration/com.apple.airport.preferences.plist`
iOS Physical: `[/private/var/]preferences/SystemConfiguration/com.apple.wifi.plist`

These plists contain network information of the "known" networks. Each network is stored in its own key.

The "remembered" networks are a list of networks the system has previously established a connection with. These items do not appear to be purged unless performed by the user.

Some of the attributes available for each network include Captive (that pop-up screen you get in hotels and restaurants), when the system last connected to the network (stored in local system time), network SSID, and automatic logon, etc.

On older macOS systems, the `com.apple.airport.preferences.plist` property list file contains similar information; however, it is organized a bit differently.

The keys under `KnownNetworks` contain keys named `wifi.ssid.<hex>`. The `<hex>` is the hex representation of the network's SSID name.

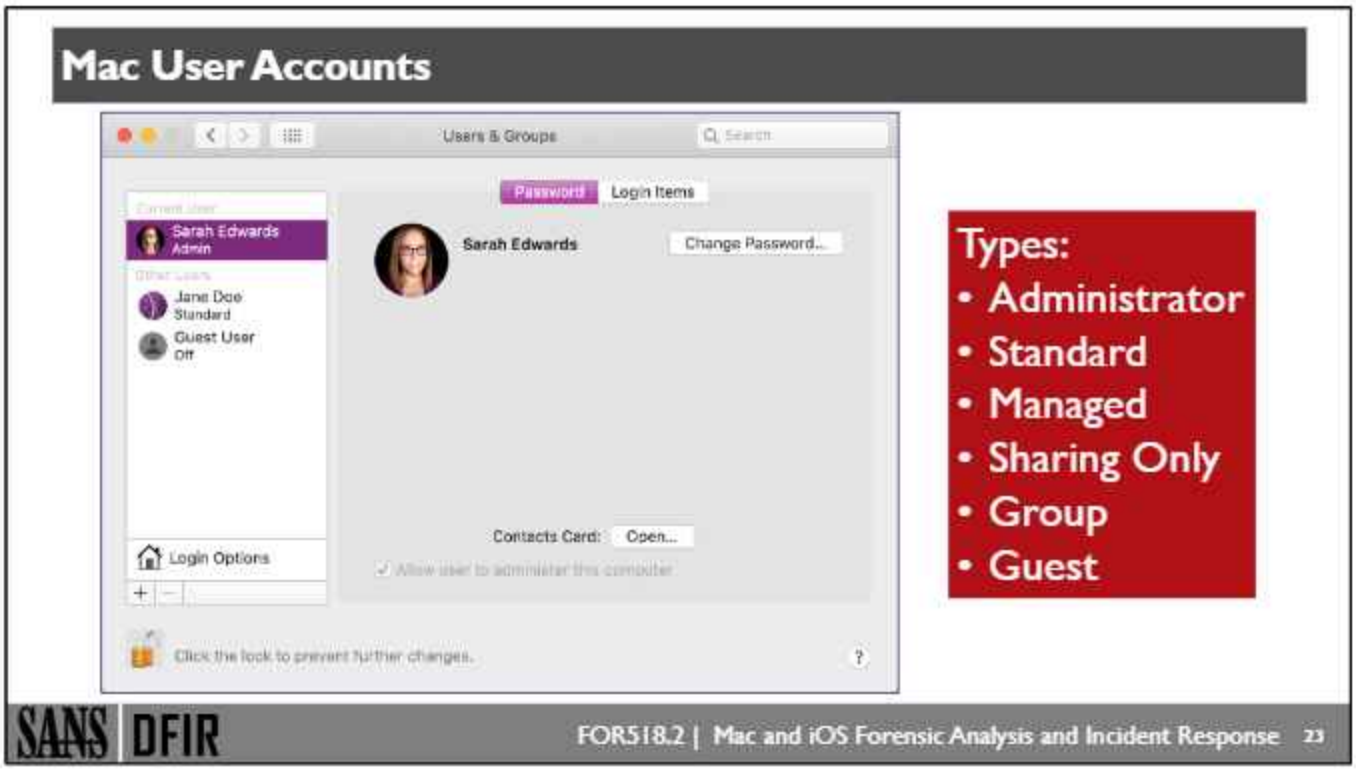
The `PreferredOrder` key contains the order in which access points should be used, in order of preference. Item 0 is the most preferred.

An example of a Wi-Fi network on iOS is the `com.apple.wifi.plist` file.

While on, Mac users can remove and modify Wi-Fi access points, but it is not as easy on iOS. Known networks are populated when iOS devices join them and are difficult to remove from the list. The user can select "Forget This Network", but only when in range of the network.

▼ List of known networks	Array	(20 items)
▼ Item 0	Dictionary	(37 items)
lastAutoJoined	Date	Dec 14, 2016, 10:50:00 AM
▶ RATES	Array	(8 items)
UserDirected	Boolean	NO
▶ networkKnownBSSListKey	Array	(2 items)
▶ RSN_IE	Dictionary	(4 items)
SCAN_RESULT_FROM_PROBE_RSP	Boolean	YES
SSID	Data	<464f5235 3138>
SSID_STR	String	FOR518
Strength	Number	0.581395328044891
80211W_ENABLED	Boolean	YES
CAPABILITIES	Number	273
BEACON_INT	Number	20
AGE	Number	21
SNR	Number	14
ASSOC_FLAGS	Number	1
▶ EXT_CAPS	Dictionary	(1 item)
ScaledRSSI	Number	0.5813953
FT_ENABLED	Boolean	YES
NOISE	Number	-88
ORIG_AGE	Number	21
PHY_MODE	Number	144
RSSI	Number	-74
FAST_ENTERPRISE_NETWORK_SUPPO...	Boolean	YES
▶ QBSS_LOAD_IE	Dictionary	(3 items)
BSSID	String	1c:b9:c4:3c:1b:c
▶ 80211D_IE	Dictionary	(1 item)
CHANNEL	Number	108
AP_MODE	Number	2
▶ VHT_CAPS_IE	Dictionary	(2 items)
ScaledRate	Number	1
networkUsage	Number	30,510.0310333371
CHANNEL_WIDTH	Number	80
IE	Data	<07125553 20240424 340
CHANNEL_FLAGS	Number	1,040
▶ HT_CAPS_IE	Dictionary	(6 items)
enabled	Boolean	YES
WiFiNetworkPasswordModificationDate	Date	Dec 9, 2016, 5:58:29 PM

Key	Type	Value
▼ Root	Dictionary	(5 items)
Counter	Number	2
▼ KnownNetworks	Dictionary	(3 items)
▶ wifi.ssid.<48796174 74204775 65737472 6f6f6d>	Dictionary	(16 items)
▶ wifi.ssid.<6d6f6269 6c652d77 6972656c 657373>	Dictionary	(15 items)
▼ wifi.ssid.<76657972 6f6e>	Dictionary	(16 items)
AutoLogin	Boolean	NO
Captive	Boolean	NO
▶ ChannelHistory	Array	(1 item)
Closed	Boolean	YES
▶ CollocatedGroup	Array	(0 items)
Disabled	Boolean	NO
LastConnected	Date	Dec 16, 2014, 5:43:48 PM
Passpoint	Boolean	NO
PossiblyHiddenNetwork	Boolean	NO
RoamingProfileType	String	Single
SPRoaming	Boolean	NO
SSID	Data	<76657972 6f6e>
SSIDString	String	veyron
SecurityType	String	WPA2 Personal
SystemMode	Boolean	YES
TemporarilyDisabled	Boolean	NO
▼ PreferredOrder	Array	(3 items)
Item 0	String	wifi.ssid.<76657972 6f6e>
Item 1	String	wifi.ssid.<6d6f6269 6c652d77 6972656c 657373>
Item 2	String	wifi.ssid.<48796174 74204775 65737472 6f6f6d>
▶ UpdateHistory	Array	(1 item)
Version	Number	2,200



Information about the user accounts is useful when trying to determine which user did what. The 'Users & Groups' preference panel shown above allows a user to add or remove users, change passwords, change their information, enable parental controls, and perform administrative functions.

Each user is associated with a particular account type:

- Administrator
- Standard
- Managed with Parental Controls
- Sharing Only
- Group
- Guest

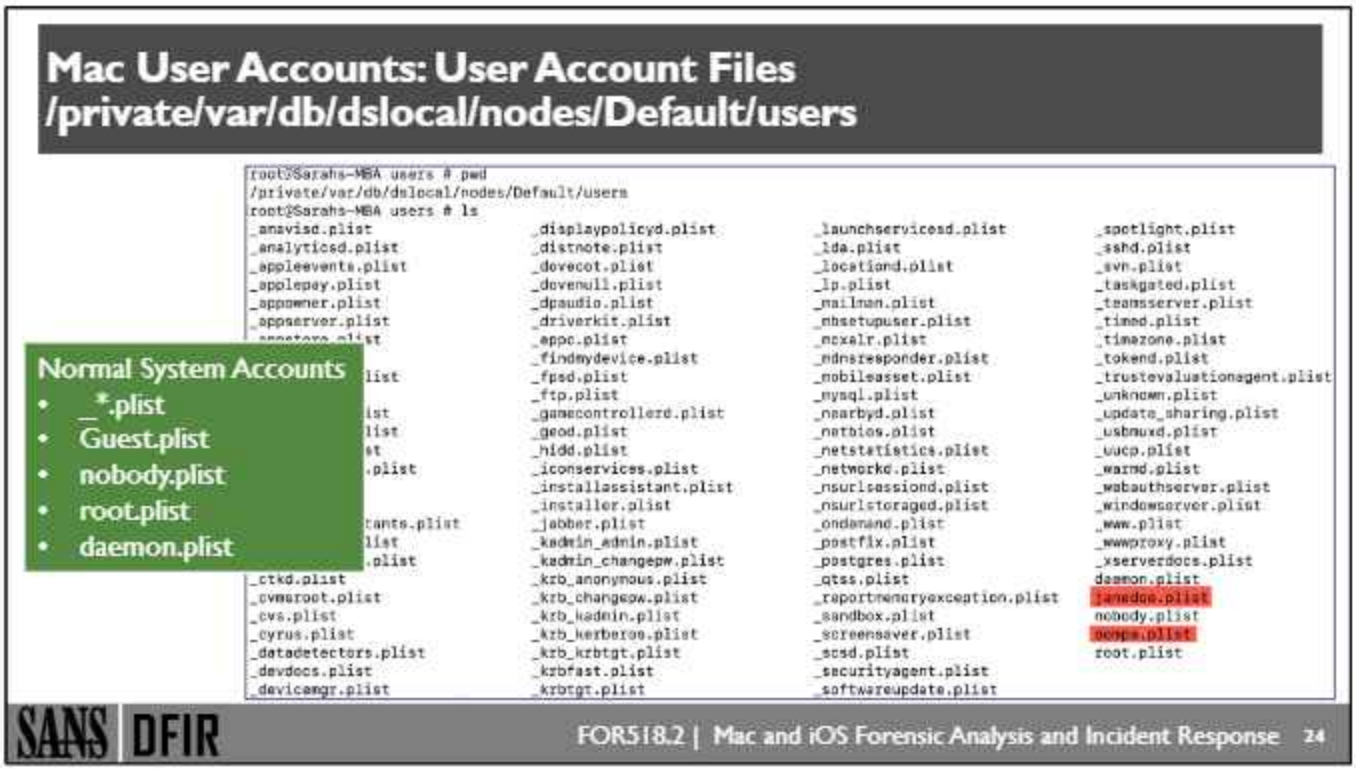
The **Administrator** has full administrative access to the system, while a **Standard** user has limited administrative privileges to the system; they can install software and change their own account preferences.

An account that is **Managed with Parental Controls** can be restricted from using certain applications, may have limited exposure to inappropriate content, or may have time usage restrictions.

A **Sharing Only** account can be used by networked users to access shared files.

A **Group** can be created to keep track of more complex user types, such as in an enterprise environment.

A **Guest user** can log in (without a password) and use the computer temporarily. If configured, a Guest user may be able to connect to shared folders or have parental controls enabled. If the system uses FileVault, the Guest user will only be able to access Safari. After a Guest user logs out, all data in the Guest home directory is deleted. Guest users on 10.7 and below are similar. One exception using 10.7: when FileVault is enabled, Guest users cannot log on.



Each user and group has a property list file containing data about their respective user. The users are located in the `/private/var/db/dslocal/nodes/Default/users` directory, while the groups are detailed in the `/private/var/db/dslocal/nodes/Default/groups` directory.

The property list files may be binary or XML, depending on the OS version.

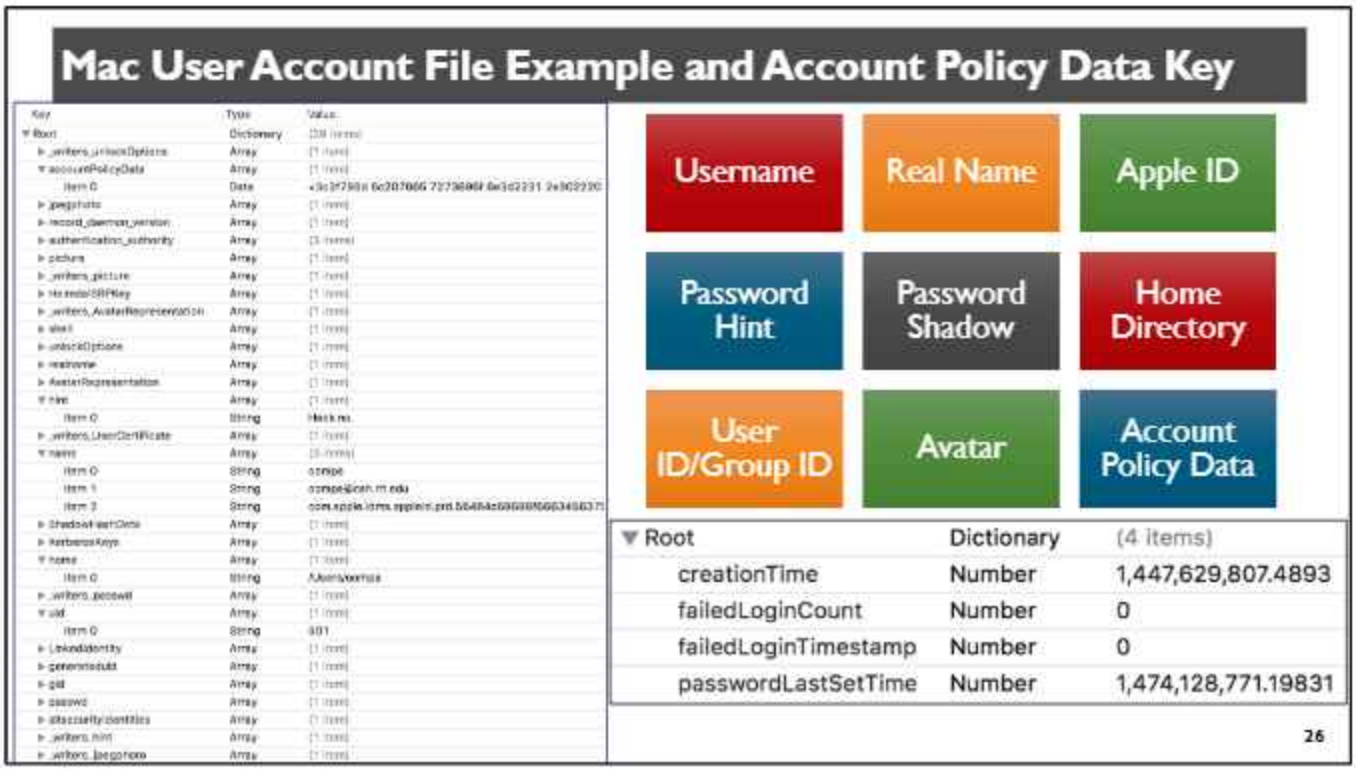
- 10.6: XML
- 10.7+: binary

Access to this directory requires root privileges.

It is worth noting that users who use Open Directory (similar to Active Directory) will not have a user plist in this directory.

The `_*.plist` are for services and groups, while `daemon.plist`, `root.plist`, and `nobody.plist` are also default accounts.

```
root@Sarahs-MBA users # pwd
/private/var/db/dslocal/nodes/Default/users
root@Sarahs-MBA users # ls
_amavisd.plist                  _displaypolicyd.plist           _launchservicesd.plist          _spotlight.plist
_analyticsd.plist               _distnote.plist                 _lda.plist                      _sshd.plist
_appleevents.plist              _dovecot.plist                  _locationd.plist                _svn.plist
_applepay.plist                 _dovenull.plist                 _lp.plist                       _taskgated.plist
_appowner.plist                 _dpaudio.plist                  _mailman.plist                  _teamsserver.plist
_appserver.plist                _driverkit.plist                _mbsetupuser.plist              _timed.plist
_appstore.plist                 _eppc.plist                     _mcxalr.plist                   _timezone.plist
_ard.plist                      _findmydevice.plist             _mdnsresponder.plist            _tokend.plist
_assetcache.plist               _fpsd.plist                     _mobileasset.plist              _trustevaluationagent.plist
_astris.plist                   _ftp.plist                      _mysql.plist                    _unknown.plist
_atsserver.plist                _gamecontrollerd.plist          _nearbyd.plist                  _update_sharing.plist
_avbdeviced.plist               _geod.plist                     _netbios.plist                  _usbmuxd.plist
_calendar.plist                 _hidd.plist                     _netstatistics.plist            _uucp.plist
_captiveagent.plist             _iconservices.plist             _networkd.plist                 _warmd.plist
_ces.plist                      _installassistant.plist         _nsurlsessiond.plist            _webauthserver.plist
_clamav.plist                   _installer.plist                _nsurlstoraged.plist            _windowserver.plist
_cmiodalassistants.plist        _jabber.plist                   _ondemand.plist                 _www.plist
_coreaudiod.plist               _kadmin_admin.plist             _postfix.plist                  _wwwproxy.plist
_coremediaiod.plist             _kadmin_changepw.plist          _postgres.plist                 _xserverdocs.plist
_ctkd.plist                     _krb_anonymous.plist            _qtss.plist                     daemon.plist
_cvmsroot.plist                 _krb_changepw.plist             _reportmemoryexception.plist    janedoe.plist
_cvs.plist                      _krb_kadmin.plist               _sandbox.plist                  nobody.plist
_cyrus.plist                    _krb_kerberos.plist             _screensaver.plist              oompa.plist
_datadetectors.plist            _krb_krbtgt.plist               _scsd.plist                     root.plist
_devdocs.plist                  _krbfast.plist                  _securityagent.plist
_devicemgr.plist                _krbtgt.plist                   _softwareupdate.plist
```



Each user property list contains data about the specific user account. It can contain a variety of different information and may sometimes be different between different versions of macOS. This is an example from a system on 10.13, where the account has been associated with an Apple ID.

The bottom screenshot shows the account policy data from the user's account file, which was extracted from the accountPolicyData key. This plist contains the account creation time and when the password was last set in Unix epoch time. The Failed Login Count and Failed Login Timestamp do not appear to be used or updated.

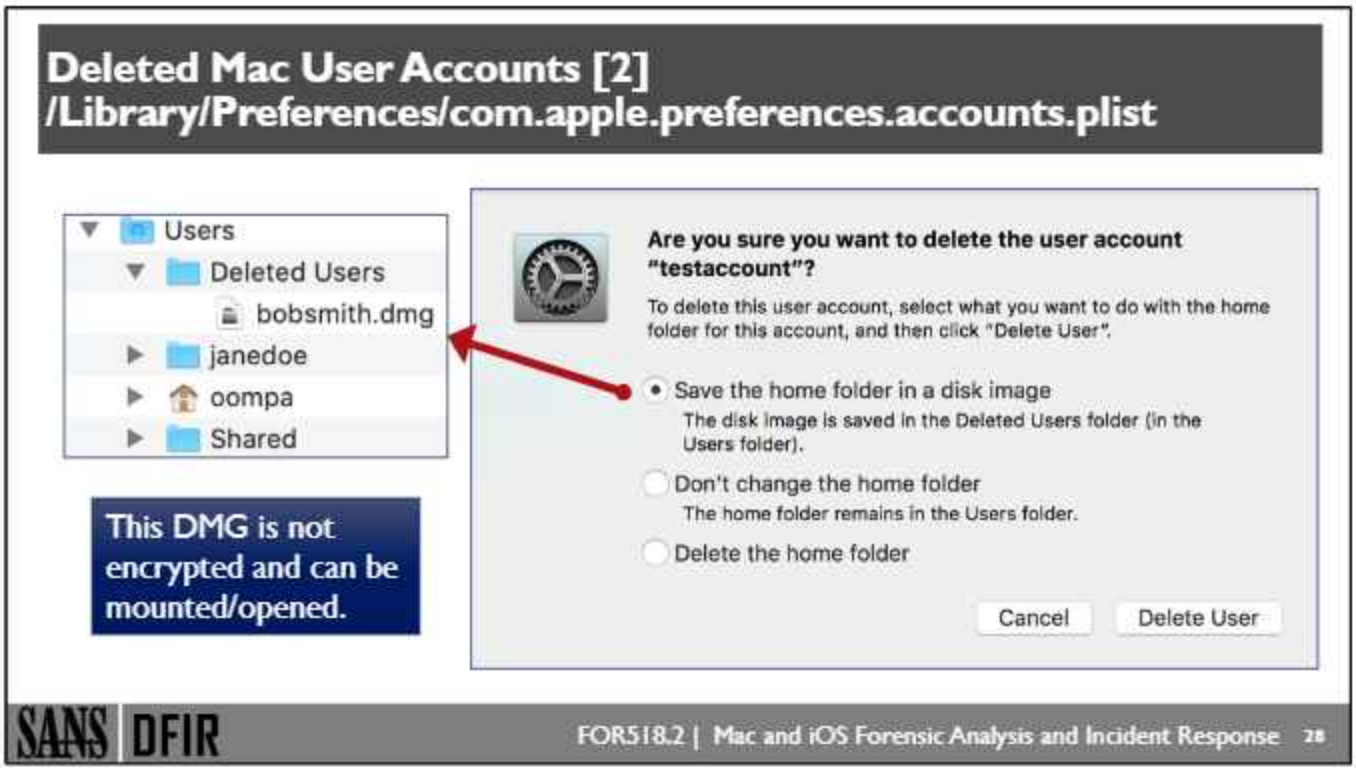
Deleted Mac User Accounts [1]
/Library/Preferences/com.apple.preferences.accounts.plist

▼ Root	Dictionary	(1 item)
▼ deletedUsers	Array	(1 item)
▼ Item 0	Dictionary	(4 items)
dsAttrTypeStandard:RealName	String	Bob Smith
date	Date	Jul 21, 2020 at 9:27:40 PM
name	String	bobsmith
dsAttrTypeStandard:UniqueID	Number	503

The deleted users are shown in the `com.apple.preferences.accounts.plist` file under the `deletedUsers` key. This property list is located in the `/Library/Preferences/` directory. If no accounts have been deleted using normal mechanisms, this plist may not exist on the system.

This key contains:

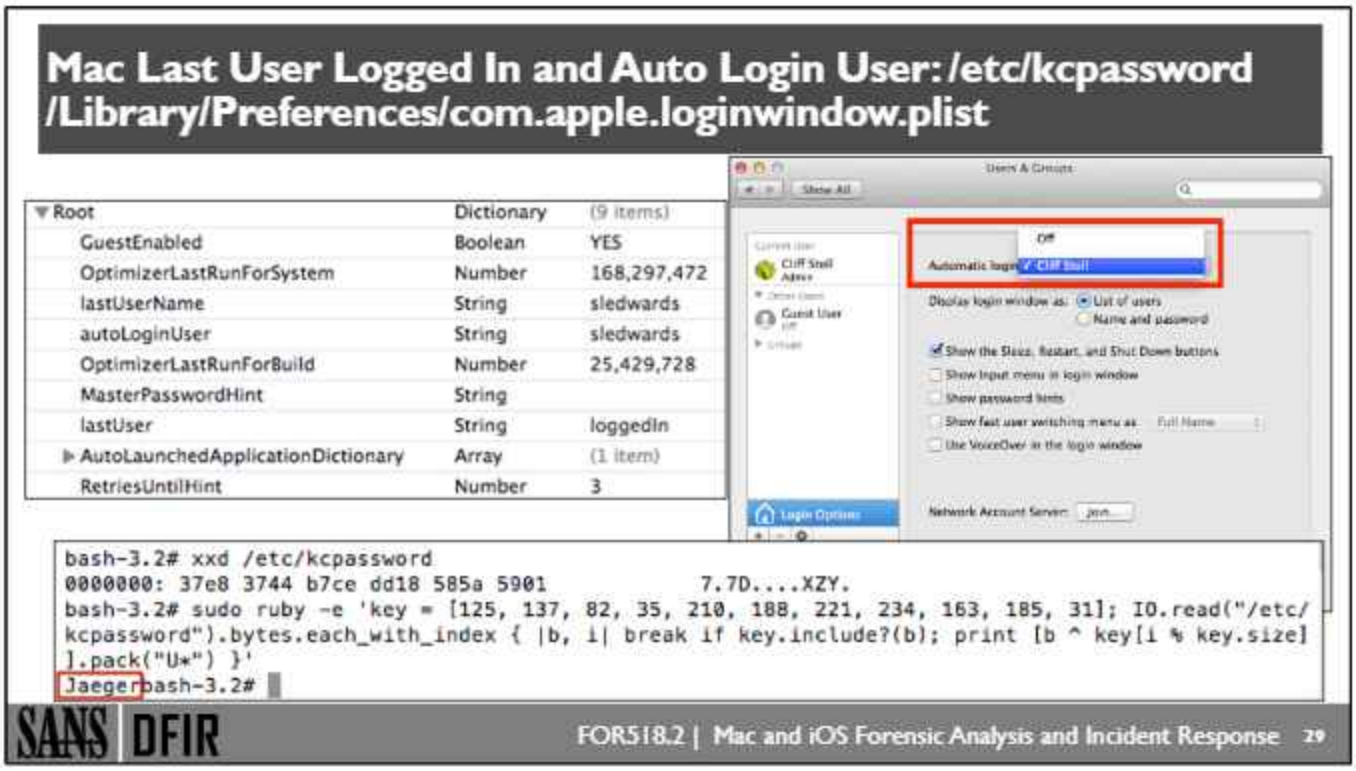
- Deleted user's "Real Name"
- UID
- Username
- Deletion date



There are three options available when a user account is deleted:

1. "`Save the home folder in a disk image`": This default option archives the user's home directory and saves it in a disk image file (DMG). The inset screenshot shows a couple of deleted user accounts, saved to DMG files, which are then moved to the `/Users/Deleted Users/` directory.
2. "`Don't change the home folder`": The home folder does not get archived; it stays in place.
3. "`Delete the home folder`": The user's home directory is removed.

When the user accounts are deleted, the user's plist in the `/private/var/db/dslocal/nodes/Default/users/` directory is also removed.



The `com.apple.loginwindow.plist` property list located in the `/Library/Preferences/` directory contains:

- `lastUser`: If the user is currently logged in (assuming the system was imaged live).
- `autoLoginUser`: The Auto Login user (if configured).
- `lastUserName`: Last logged in user.
- `RetriesUntilHint`: The number of times before a password hint is given.
- `GuestEnabled`: Guest Account Status.
- `MasterPasswordHint`: If a master password has been configured, a hint may have also been configured.

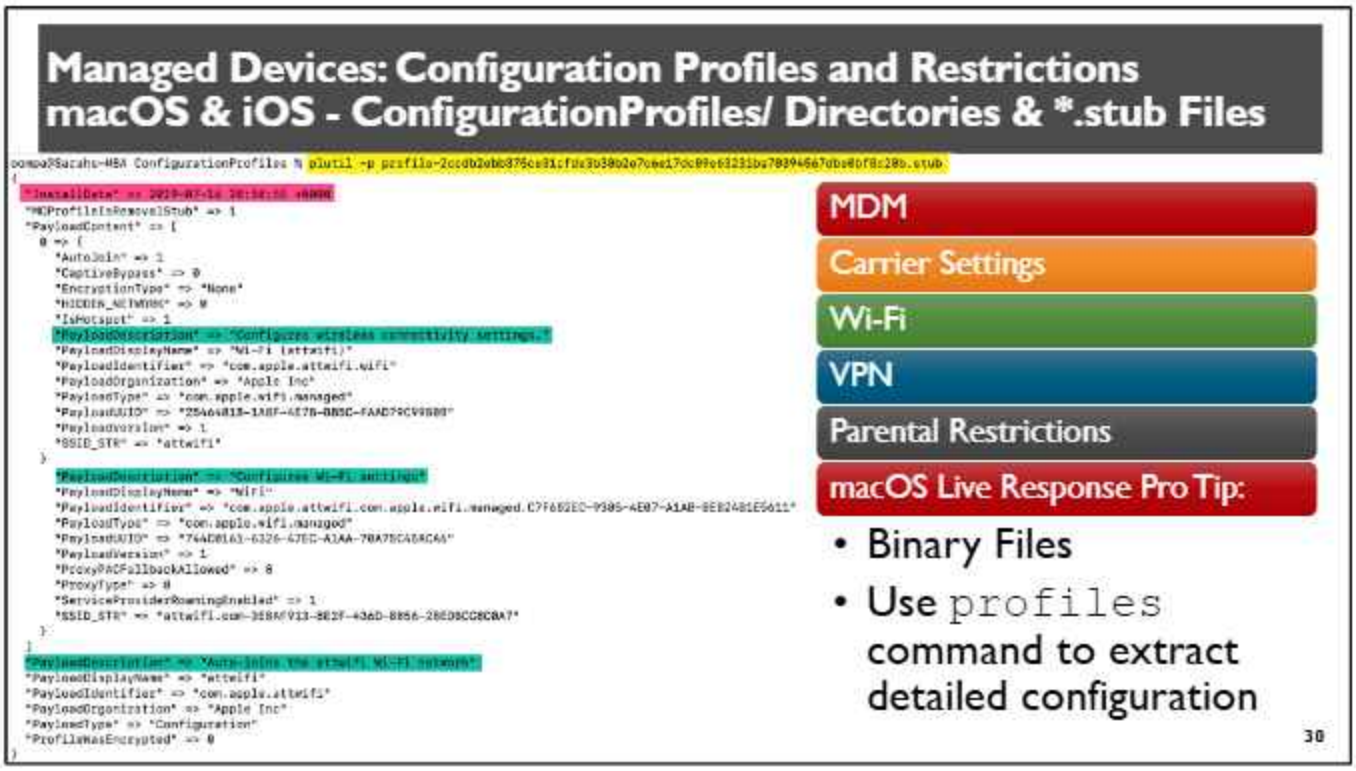
A user may choose the ability to have the system automatically log them on using the "`Automatic login`" selection window in the 'Users & Groups' preferences pane.

The user's password is then XOR'd with a multi-byte key and stored in `/etc/kcpassword`. To decode this password, use the Ruby script below from https://gist.github.com/opshope/32f65875d45215c3677d.

```
sudo ruby -e 'key = [125, 137, 82, 35, 210, 188, 221, 234, 163, 185, 31];
IO.read("/etc/kcpassword").bytes.each_with_index { |b, i| break if
key.include?(b); print [b ^ key[i % key.size]].pack("U*") }'
```

Automatic login is not available for a system's User FileVault or accounts that use iCloud credentials to log in. It may be possible to find this file on the system even after Automatic login was turned off, however.

References:
https://gist.github.com/opshope/32f65875d45215c3677d
https://apple.stackexchange.com/questions/50652/does-activating-auto-login-compromise-secure-password-storage
https://support.apple.com/en-us/HT201476



macOS: /private/var/db/ConfigurationProfiles/
iOS: `/private/var/mobile/Library/ConfigurationProfiles/`,
/private/var/mobile/Library/UserConfigurationProfiles/,
/private/var/containers/Shared/SystemGroup/systemgroup.com.apple.configurationprofiles,
./containers/Shared/SystemGroup/systemgroup.com.apple.configurationprofiles

Devices can be managed (or supervised). They may be managed through your enterprise Mobile Device Management system or settings pushed to you by your corporation or carrier (as well as many other ways).

Managed devices will have a "profile" installed on the device. This profile can be reviewed to see what actions/limitations it can use. These profiles are found in the ConfigurationProfiles directory with a filename similar to, "`profile-<hash>.stub`". The example above shows a standard profile on AT&T devices that allows these devices to auto-join "attwifi" access points. These profiles can be accessed through the GUI in the following path: `Settings | General | Profiles` (or `Device Management`). Note: Hidden profiles, like the AT&T one above, will not show up, as it is configured as a "hidden" profile. If no "non-hidden" profiles are installed, the menu option does not exist.

Restrictions (`Settings | General | Restrictions`) may also be used to limit certain applications, services, app purchases, media content, privacy restrictions, etc. This is similar to parental-type controls found on other devices and systems. Devices that have restrictions may have those restrictions recorded in the `UserSettings.plist`, `EffectiveUserSettings.plist`, and `PublicEffectiveUserSettings.plist` files in the `ConfigurationProfiles` directory.

Reference:
https://developer.apple.com/business/documentation/Configuration-Profile-Reference.pdf

```
oompa@Sarahs-MBA ConfigurationProfiles % plutil -p profile-2ccdb2ebb375ce31cfde3b30b2e7c6e17dc09e63231ba70394567dbe0bf8c20b.stub
{
  "InstallDate" => 2019-07-16 20:50:55 +0000
  "InstallOptions" => {
    "filterFlag" => 18
    "isInstalledInteractively" => 0
    "signatureVersion" => 2
  }
  "MCProfileIsRemovalStub" => 1
  "PayloadContent" => [
    0 => {
      "AutoJoin" => 1
      "CaptiveBypass" => 0
      "EncryptionType" => "None"
      "HIDDEN_NETWORK" => 0
      "IsHotspot" => 1
      "PayloadDescription" => "Configures wireless connectivity settings."
      "PayloadDisplayName" => "Wi-Fi (attwifi)"
      "PayloadIdentifier" => "com.apple.attwifi.wifi"
      "PayloadOrganization" => "Apple Inc"
      "PayloadType" => "com.apple.wifi.managed"
      "PayloadUUID" => "25464818-1A8F-4E78-886C-FAAD79C99B0B"
      "PayloadVersion" => 1
      "ProxyPACFallbackAllowed" => 0
      "ProxyType" => 0
      "ServiceProviderRoamingEnabled" => 0
      "SSID_STR" => "attwifi"
    }
    1 => {
      "AutoJoin" => 1
      "CaptiveBypass" => 0
      "DisplayedOperatorName" => "AT&T Wi-Fi Passpoint"
      "EncryptionType" => "WPA2"
      "HIDDEN_NETWORK" => 0
      "IsHotspot" => 1
      "MCCAndMNCs" => [
        0 => "310410"
      ]
      "PayloadDescription" => "Configures Wi-Fi settings"
      "PayloadDisplayName" => "WiFi"
      "PayloadIdentifier" => "com.apple.attwifi.com.apple.wifi.managed.C7F682EC-9305-4E07-A1AB-BE824B1E5611"
      "PayloadType" => "com.apple.wifi.managed"
      "PayloadUUID" => "744C8161-6326-47EC-A1AA-70A75C45ACA6"
      "PayloadVersion" => 1
      "ProxyPACFallbackAllowed" => 0
      "ProxyType" => 0
      "ServiceProviderRoamingEnabled" => 1
      "SSID_STR" => "attwifi.com-3E8AF913-8E2F-436D-8856-28EDBCC8C0A7"
    }
  ]
  "PayloadDescription" => "Auto-joins the attwifi Wi-Fi network"
  "PayloadDisplayName" => "attwifi"
  "PayloadIdentifier" => "com.apple.attwifi"
  "PayloadOrganization" => "Apple Inc"
  "PayloadType" => "Configuration"
  "PayloadUUID" => "8B808F99-2FCF-4110-8E1C-697D7988EC81"
  "PayloadVersion" => 1
  "ProfileWasEncrypted" => 0
  "TargetDeviceType" => 0
}
```

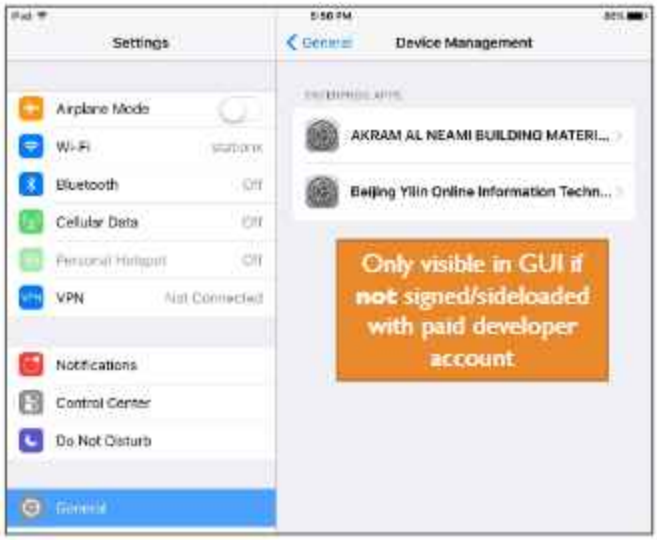
iOS (& macOS) Managed Devices: Enterprise Provisioning Profiles /MobileDevice/ProvisioningProfiles/

Enterprise deployments

Allows apps to be run without downloading from Apple App Store

Malware on non-jailbroken devices (WireLurker)

Jailbroken devices (Pangu (shown), unc0ver, etc.)



SANS DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 32

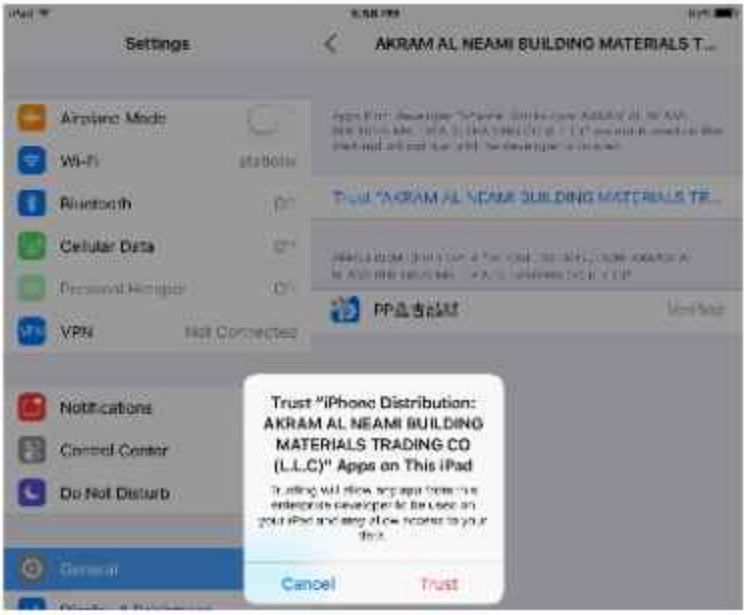
macOS: /private/var/db/ConfigurationProfiles/
iOS: `/private/var/MobileDevice/ProvisioningProfiles/`

Devices can run applications distributed by an enterprise using a Provisioning Profile. These profiles allow applications to be distributed and run by users without having to download them from the Apple App Store. Devices do not need to be jailbroken to run these applications. These profiles allow applications to be run by trusting a certificate that is downloaded by the user onto the device.

These certificates are stored in the ProvisioningProfiles directories as files with a GUID-based filename. These files contain information such as the enterprise name, bundle ID, creation timestamp, developer certificate, and entitlements.

References:

- https://www.paloaltonetworks.com/content/dam/pan/en_US/assets/pdf/reports/Unit_42/unit42-wirelurker.pdf
- (Dev Account Required) https://help.apple.com/developer-account/#/



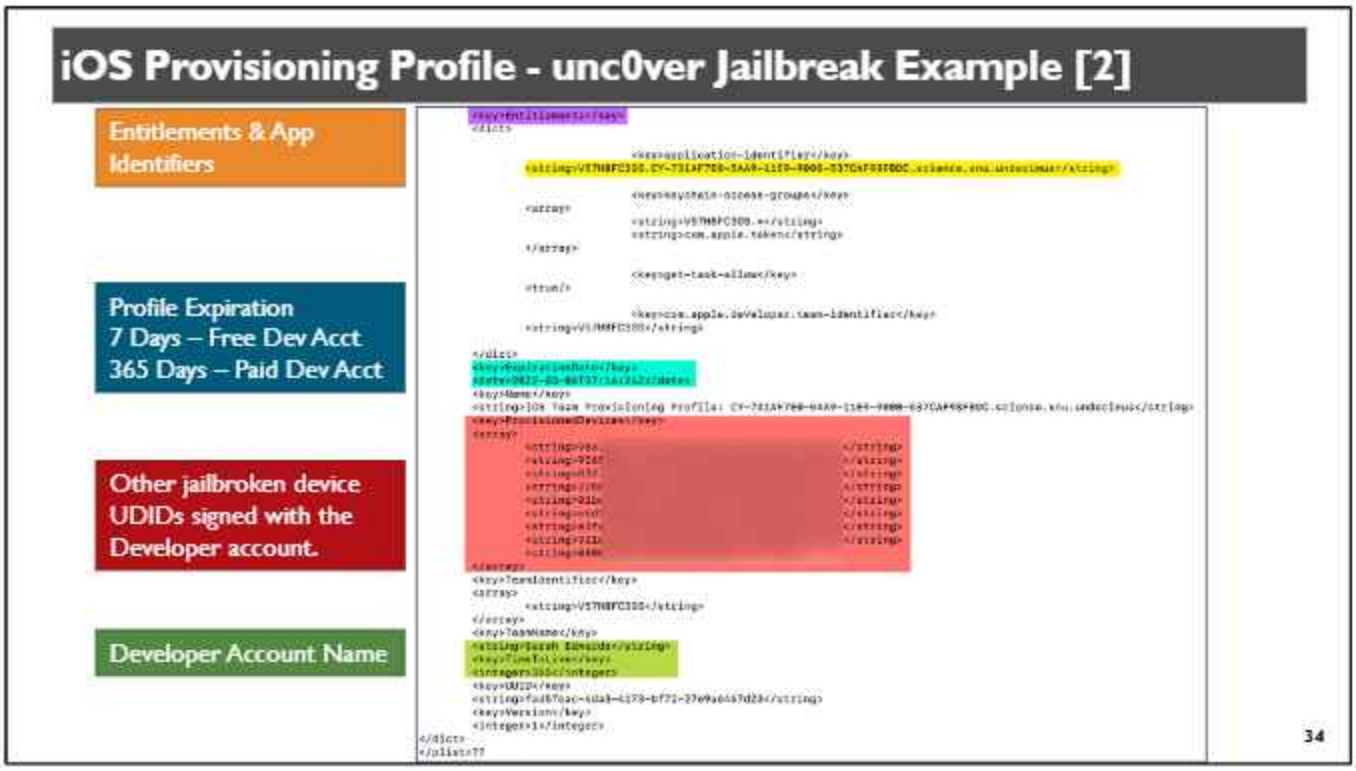
iOS Provisioning Profile - unc0ver Jailbreak Example [1]

```
oompa@MBP-M1 ProvisioningProfiles % pwd
/Users/oompa/miphoneX_14_2_unc0ver/private/var/MobileDevice/ProvisioningProfiles
oompa@MBP-M1 ProvisioningProfiles % head -n 25 fad5fcac-4da3-4173-bf72-27e9a6467d23
??.?0?.?1?H??
??4?0<?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>AppIDName</key>
        <string>CY- unc0ver</string>
        <key>ApplicationIdentifierPrefix</key>
        <array>
        <string>V57H8FC3GS</string>
        </array>
        <key>CreationDate</key>
        <date>2021-03-06T17:16:24Z</date>
        <key>Platform</key>
        <array>
                <string>iOS</string>
        </array>
        <key>IsXcodeManaged</key>
        <true/>
        <key>DeveloperCertificates</key>
        <array>
                <data>MIIFsjCCBJqgAwIBAgIQWOZymIRJklct8C1Qif4BGTANBgkqhkiG9w0BAQsFADB1MUQwQgYDVQQDDDtBcHBs
ZSBXb3JsZHdpZGUgRGV2ZWxvcGVyIFJlbGF0aW9uc2hpcyBDZXJ0aWZpY2F0aW9uIEF1dGhvcml0eTELMAkGA1UECwwCRzMxEzARBgNVBAoMCk
```


Provisioning profiles have a GUID-based filename. This is a binary file that contains a plist (shown) and certificate information.

The contents of the plist can be useful to determine if these profiles are suspicious or legitimate. Look for App names (unc0ver) and timestamps and developer certificates. The `creationDate` is when the application was sideloaded to the device.

This plist is continued on the next slide.

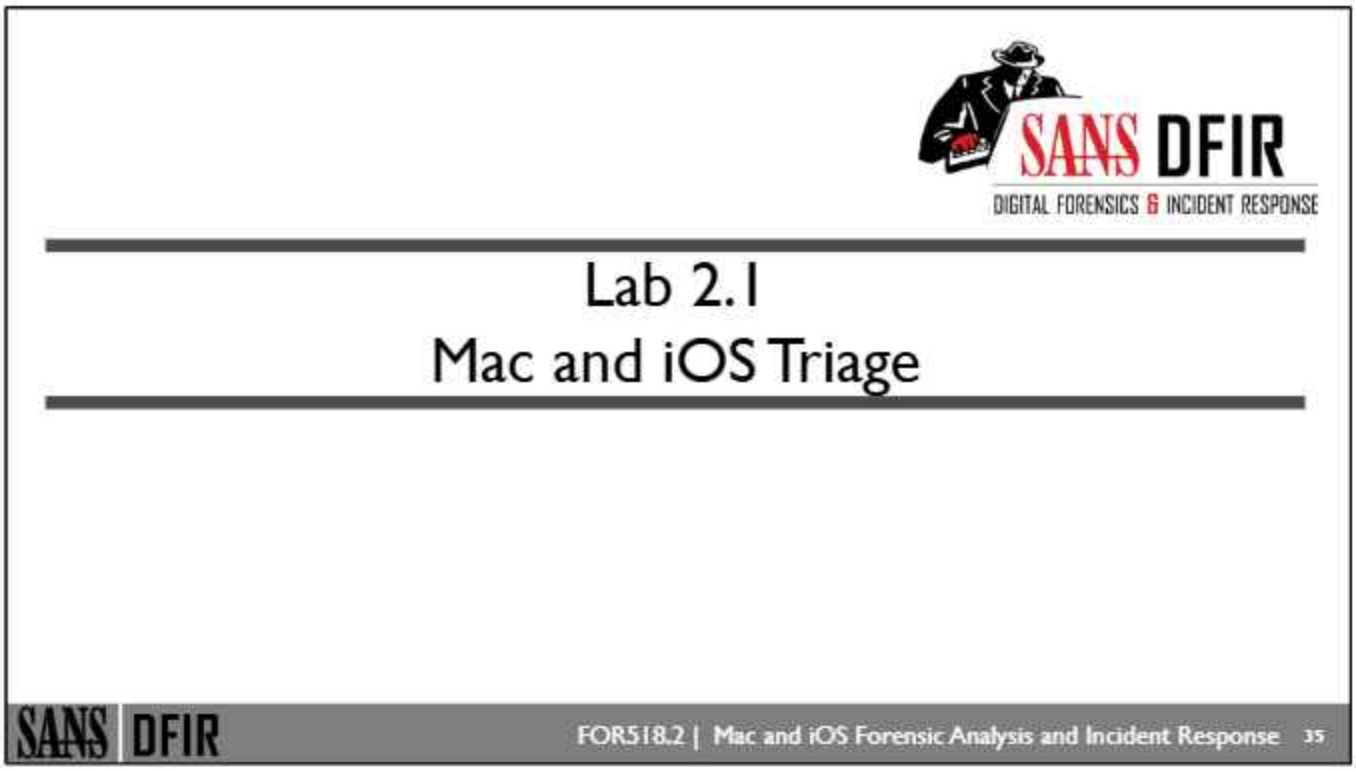


The profile plist also contains app entitlements, and more app identifiers.

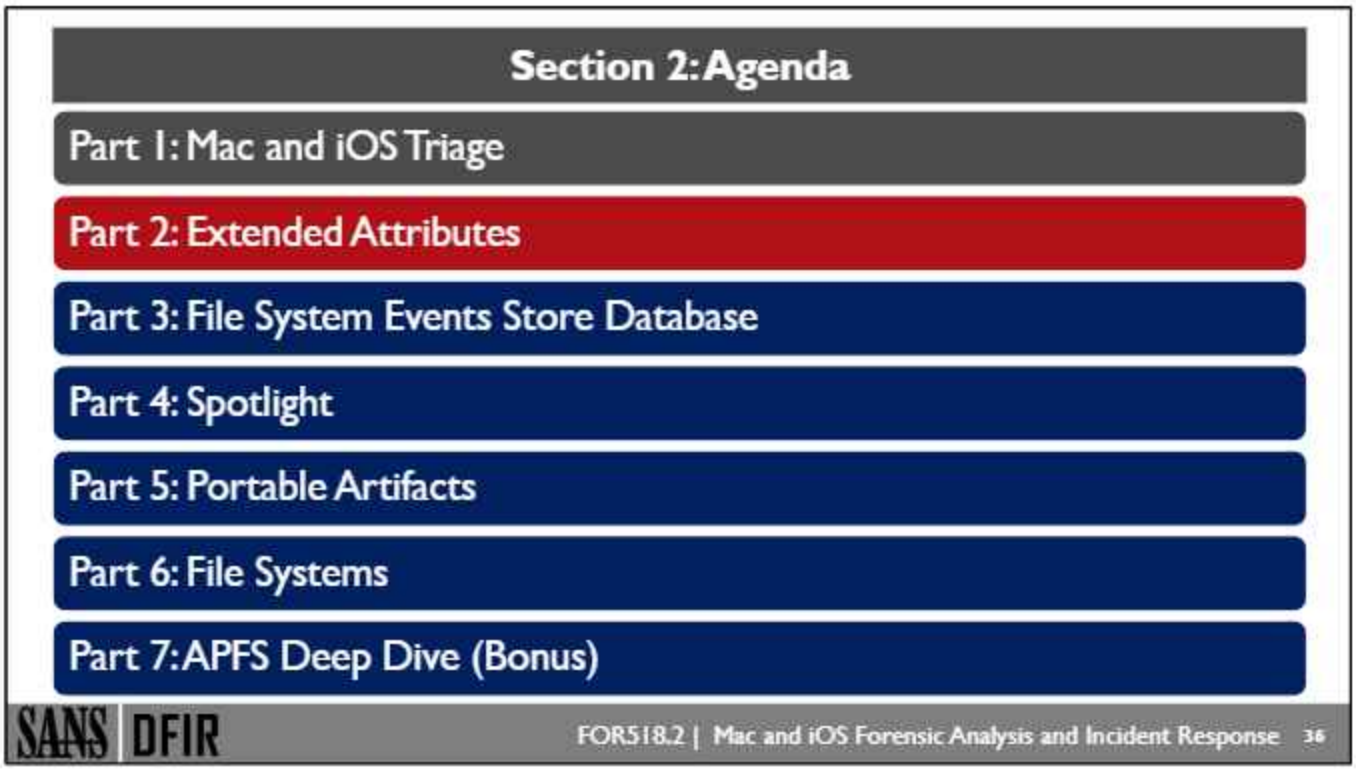
Take note of the `ExpirationDate`. Applications that were sideloaded without a paid developer account will expire after 7 days, while one with a paid developer account can last for 365 days (and will not appear in the GUI).

Under the `ProvisioningDevices`, key lists all the device UDIDs that also have this application. (I have quite a few jailbroken devices on hand)!

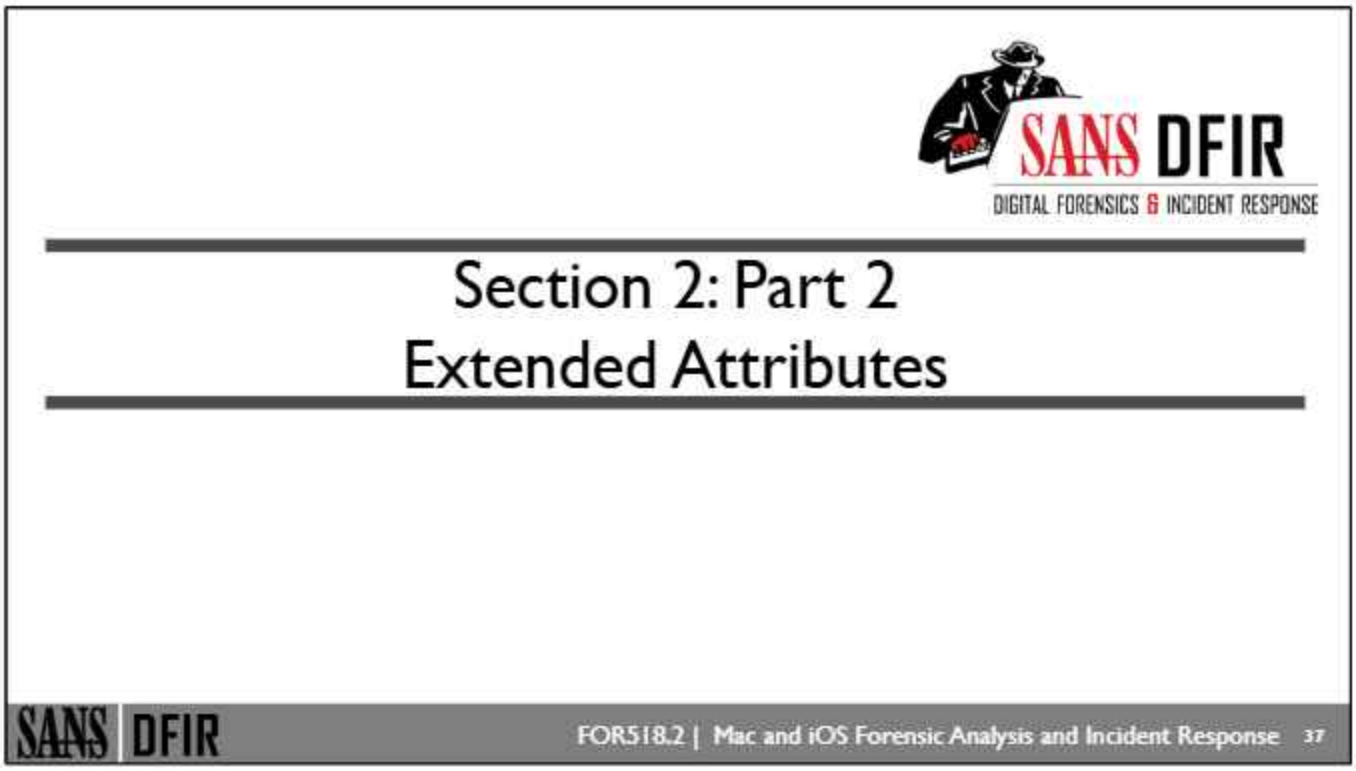
The developer account which provided the signing is also shown (Sarah Edwards) along with the Time To Live (TTL) for the profile.



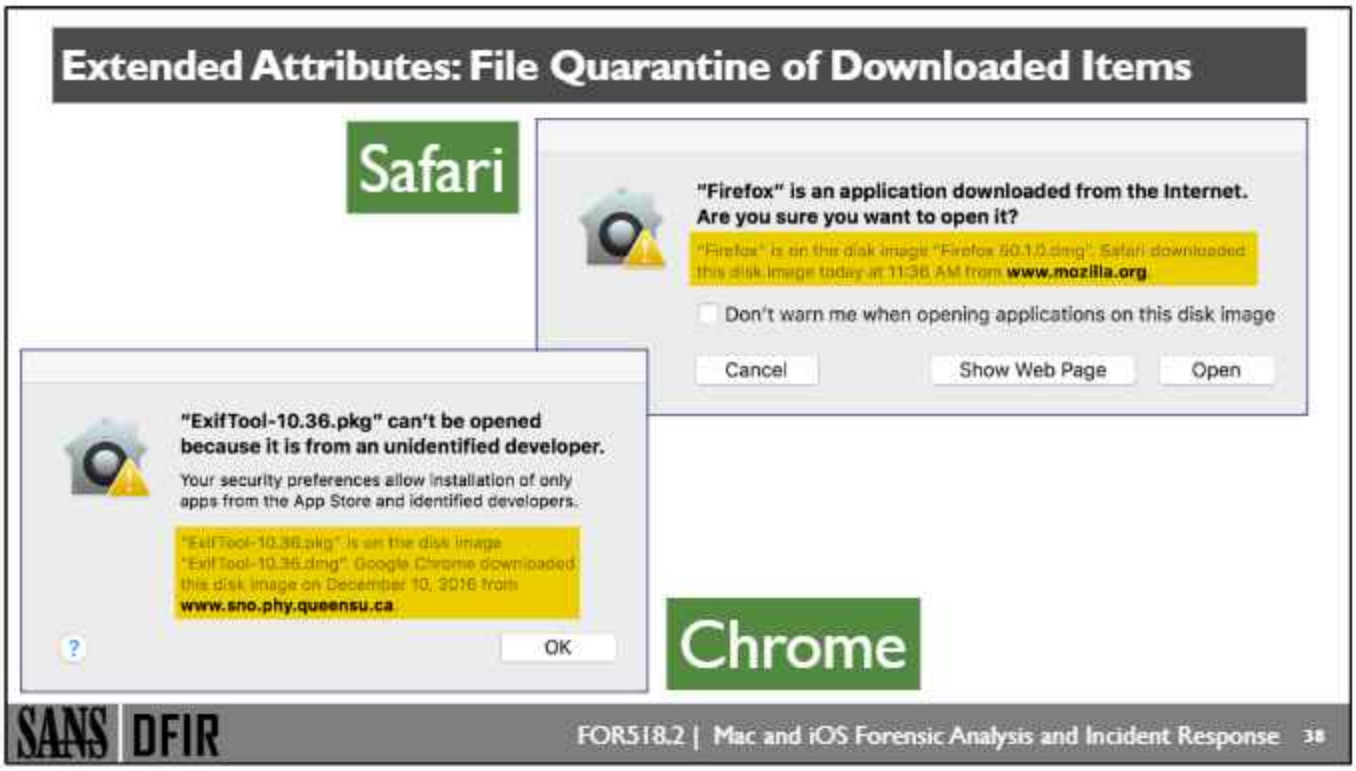
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.

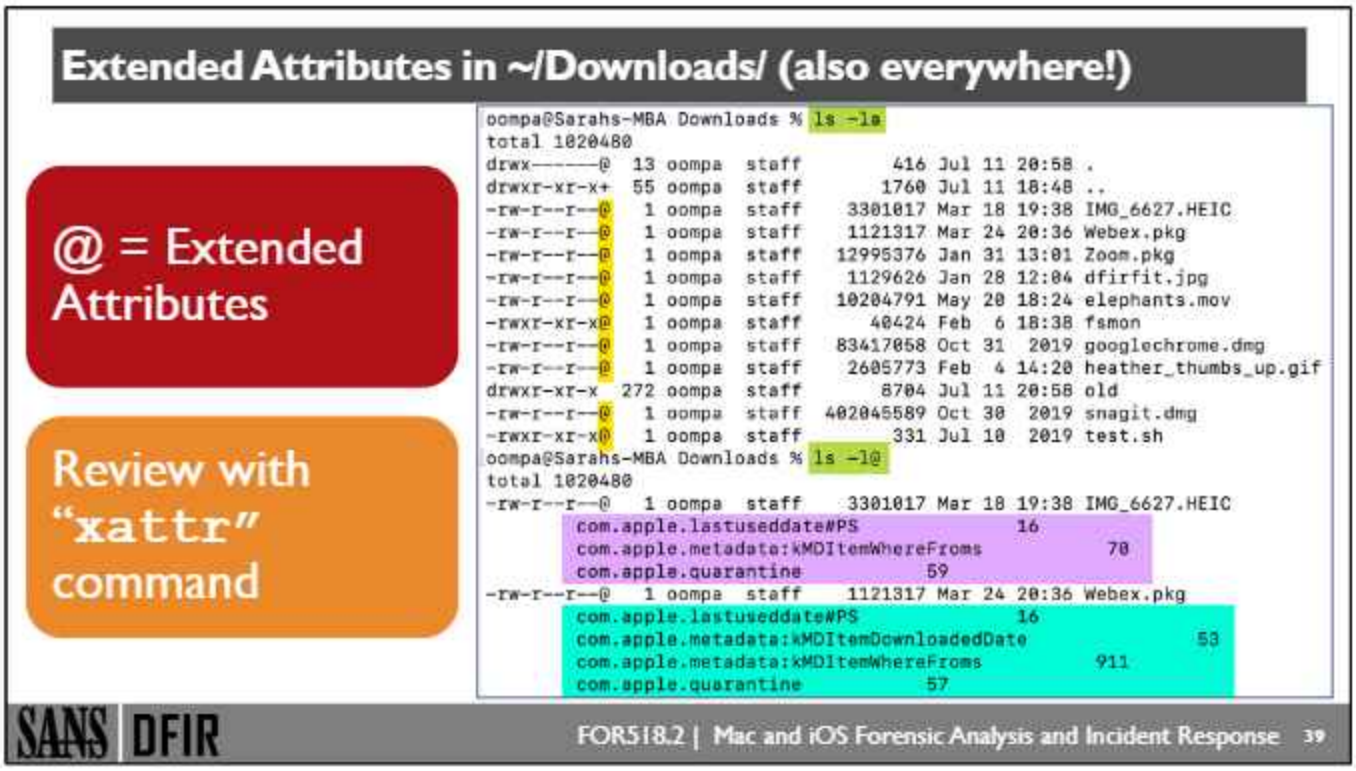


Apple uses file quarantine to protect the system by checking files against a malicious file database (more on this later). It is also used to inform the user when and where a file was downloaded from when the user attempts to open the file.

Two examples are shown above. The top example shows a file downloaded using the Safari web browser. The file downloaded was "`Firefox 50.1.0.dmg`", which was downloaded at 11:36 AM from mozilla.org.

The bottom example shows that "`ExifTool-10.36.pkg`" was downloaded using the Chrome web browser on December 10, 2016, from "www.sno.phy.queensu.ca".

How does the Mac know when and where these files were downloaded from? … Extended attributes!

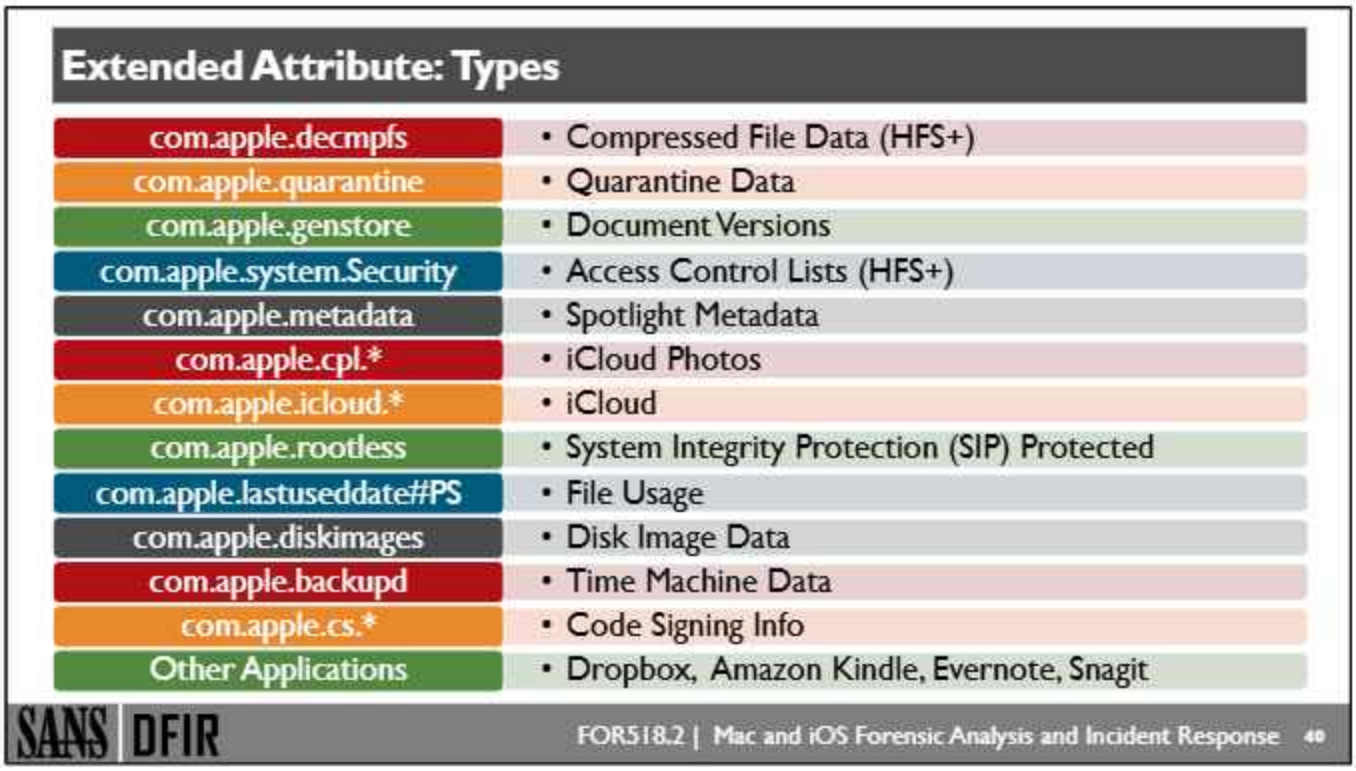


Each user has their own `Downloads` directory. This directory contains downloads from applications, such as web browsers and email that use this folder as their default download location. Most applications will have a preference setting that users can change; however, by default, this directory will be used.

On a well-used system, this directory will contain months, potentially even years' worth of downloads.

Each item that was downloaded will likely have extended attributes showing metadata of the download. To see if a file contains extended attributes, you can run the command "`ls -la`".

Files with extended attributes will have the "`@`" at the end of the permissions. You can even try running the command "`ls -l@`" to see the names of the attributes.

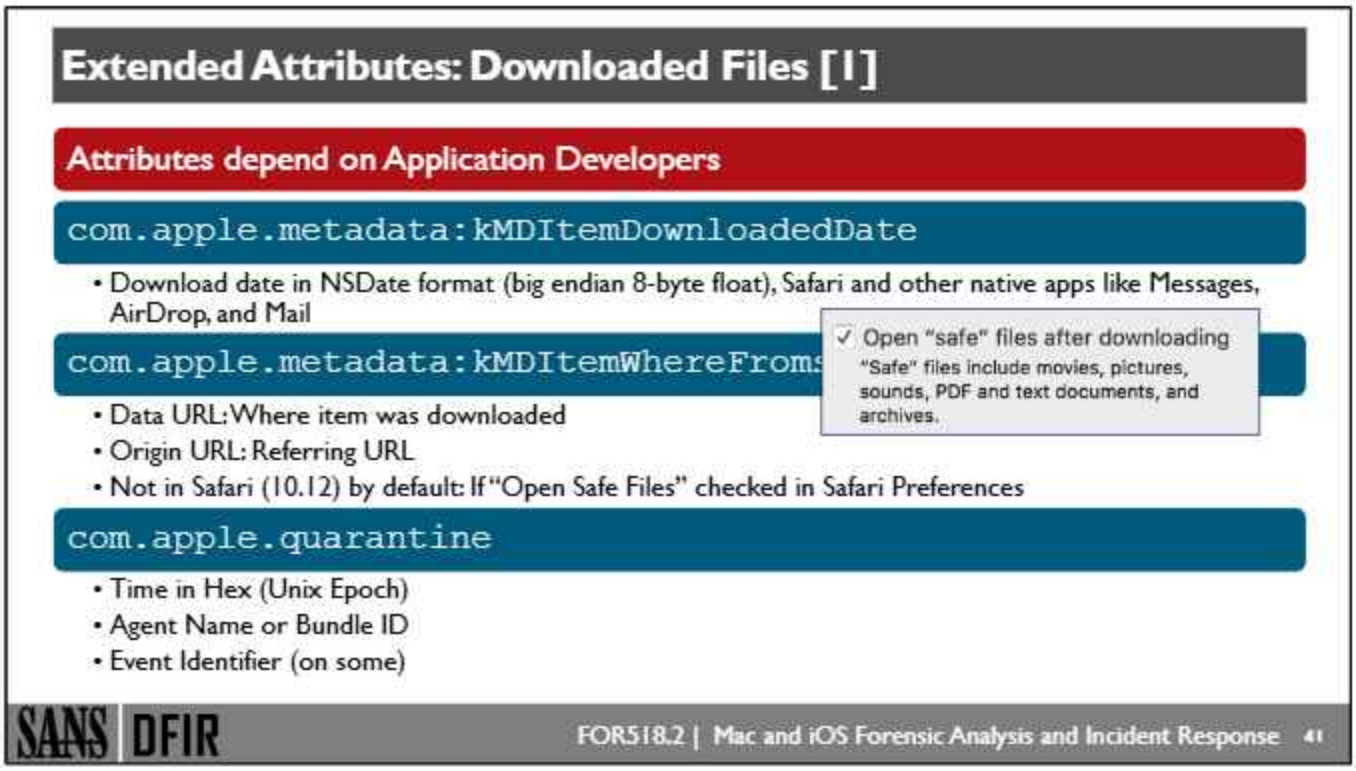


Extended attributes come in many types, some of which are listed above. The two most common are `com.apple.decmpfs` and `com.apple.quarantine`. Each attribute contains data specific to its purpose and does not follow a static format. An investigator's intuition and reverse engineering skills may be needed to decode the contents of these attributes.

The `com.apple.decmpfs` attribute is particularly interesting because it means this file is using the per-file compression available in the HFS+ file system. Files such as those found in `/bin`, like `ls`, `echo`, or `dd` are compressed.

Ever wonder why some versions of EnCase show these files as having 0 bytes? This is why.

Note: The `xattr` command filters out `com.apple.decmpfs` and `com.apple.system.Security` attributes. An investigator may need additional tools to review these two attributes.

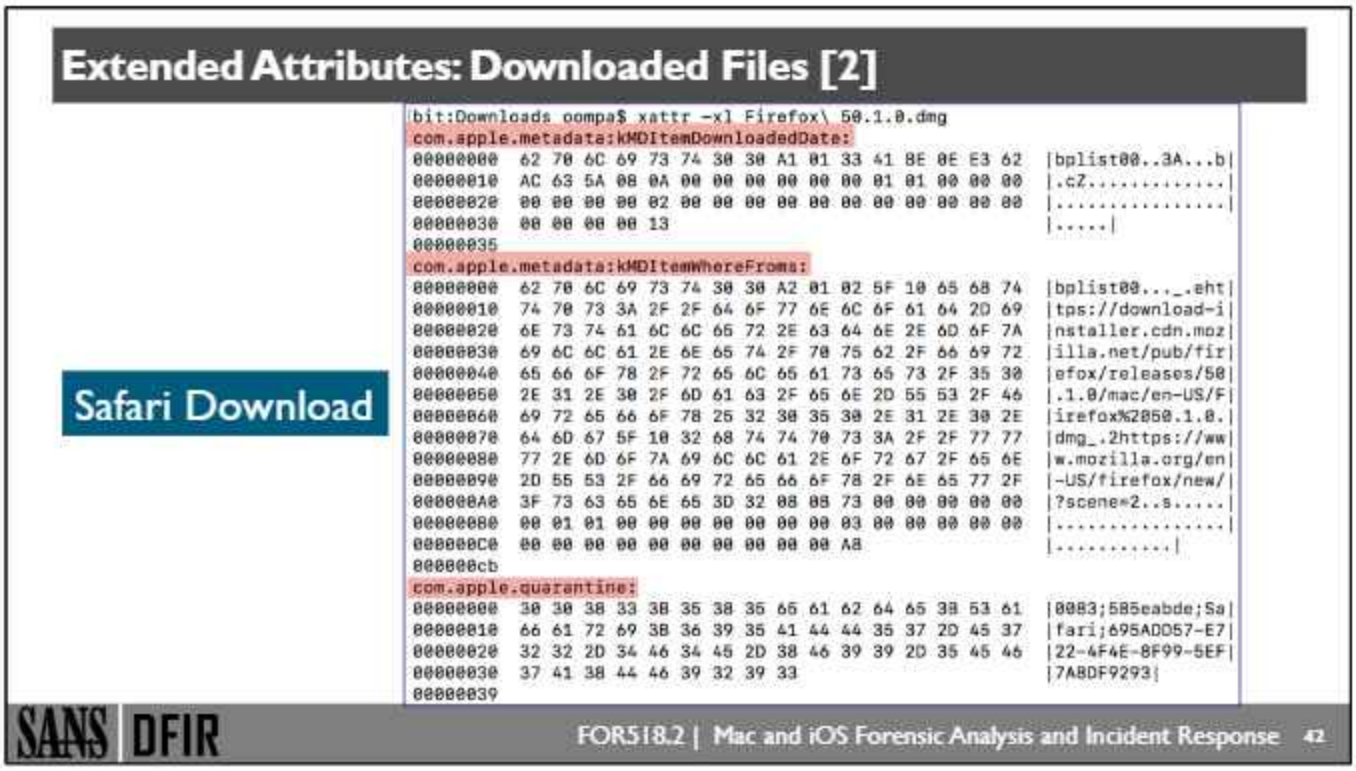


The contents of extended attribute data can vary by operating system or application. Each browser saves different extended attributes.

Safari: All in slide
Chrome: Does not use `kMDItemDownloadedDate`

The extended attributes can contain useful information, such as:

- Download dates.
- URLS: The Data URL shows where the data was actually downloaded from. The Origin URL shows where the download was referred from. This extended attribute will not get written for some of Safari's downloaded files on 10.12 if the "Open Safe Files" option is checked in Safari Preferences.
- Quarantine Data: Contains information related to Apple's file quarantine system.

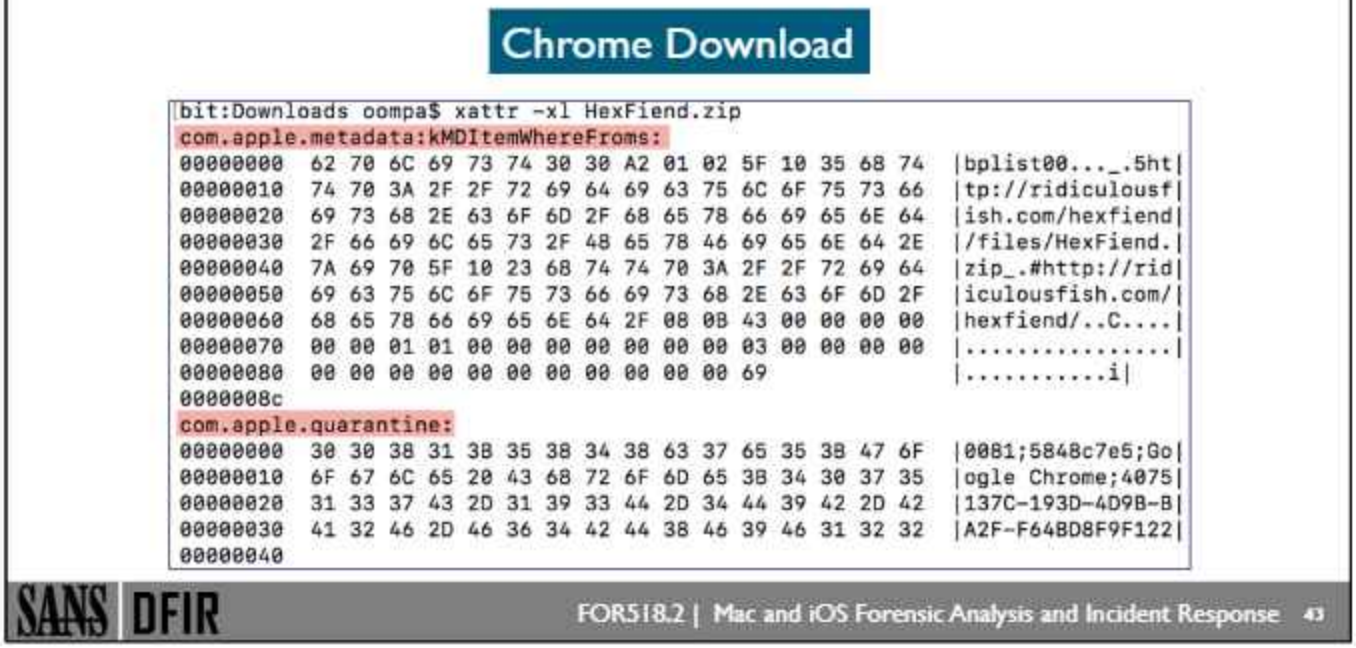


Similar to the previous example, this shows one more extended attribute that gets written when a file is downloaded using the Safari browser.

The "`com.apple.metadata:kMDItemDownloadedDate`" attribute contains a binary plist file that holds the download date.

Another browser, Firefox, will only show the "`com.apple.quarantine`" attribute information; it does not write the other two. Different applications can use different extended attributes.

Extended Attributes: Downloaded Files [3]



A file or directory on macOS may have additional metadata called extended attributes. These attributes were introduced in 10.4 as a way to incorporate more metadata functionality into macOS.

Extended attributes are stored as inline attributes in the HFS+ Attributes File.

Extended attributes can contain a variety of information. The attributes mainly found on files downloaded with the Chrome browser include:

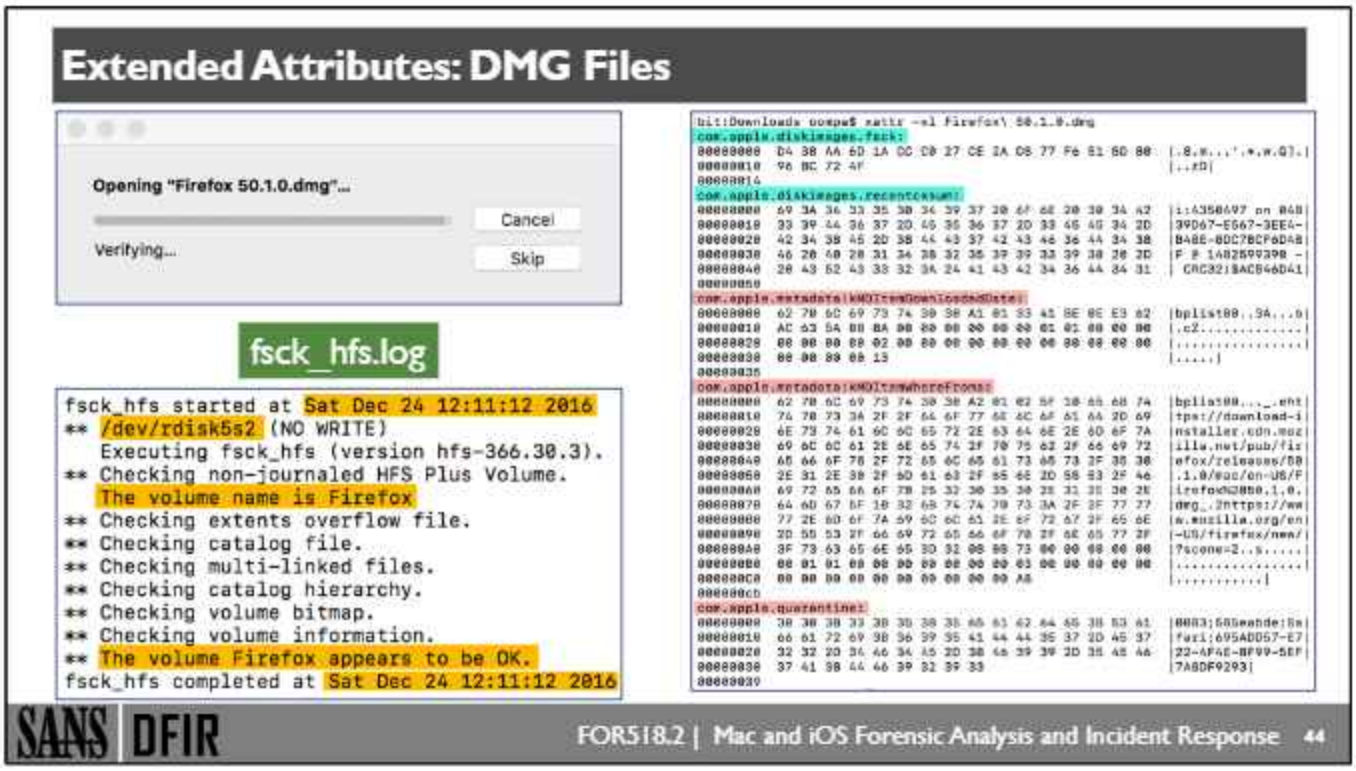
- `com.apple.metadata:kMDItemWhereFroms:` Where the file was downloaded from and the referring URL
- `com.apple.quarantine:` When the file was downloaded (hex Unix epoch value) and what application it was downloaded with

Extended attributes can be viewed by using the `xattr` command shown above.

The `xattr` command has the following output options:

- `-x:` View in hex
- `-l:` Outputs the attribute name and contents

Reference:
Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin—Chapter 16: HFS+ File System Concepts

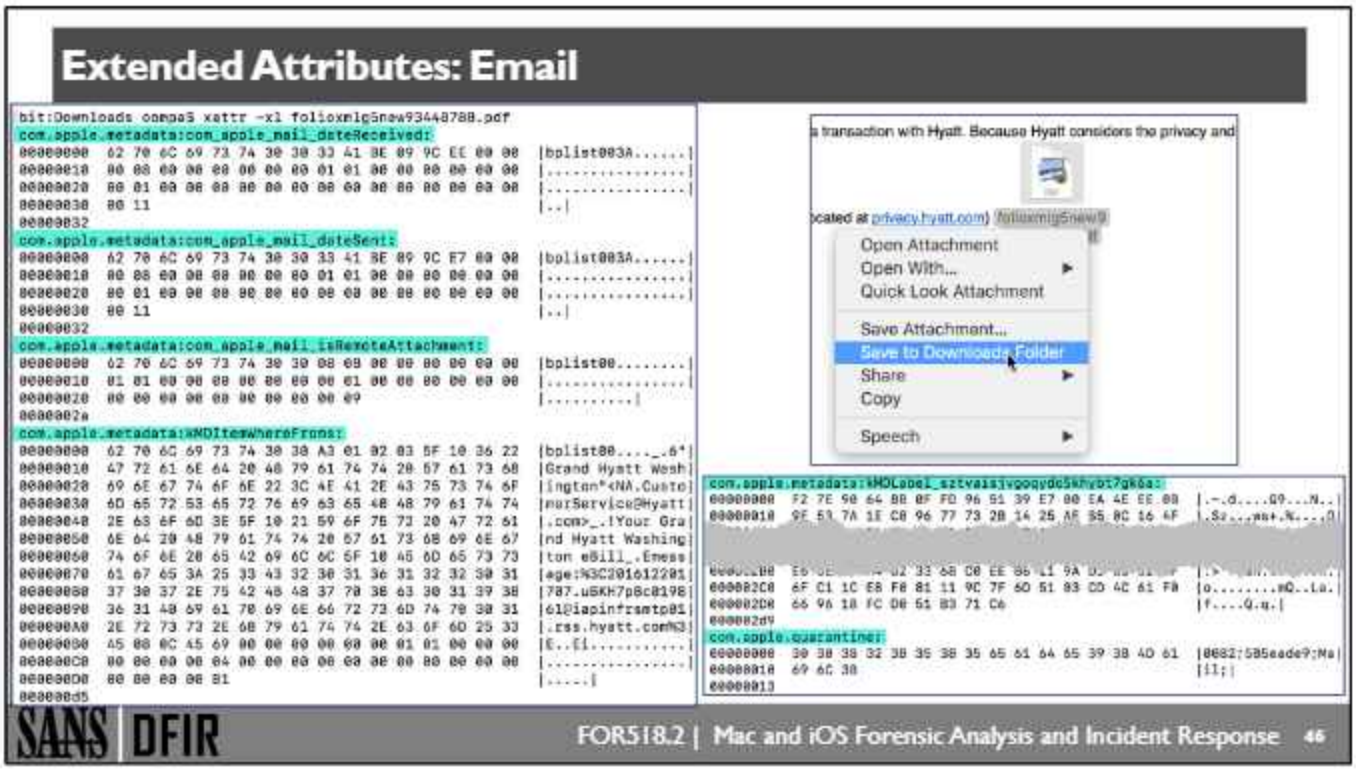


This screenshot shows another type of extended attribute specific to DMG files. Once a DMG file has been double-clicked and opened, two extended attributes get written.

- `com.apple.diskimages.fsck`: File System Check information
- `com.apple.diskimages.recentcksum`: Checksum information, including a Unix epoch timestamp of when the file was downloaded

These additional extended attributes show that a DMG was actually opened at least once and not just downloaded. The first open timestamp from this process can be found in the user's `~/Library/Logs/fsck_hfs.log` file.

```
bit:Downloads oompa$ xattr -xl Firefox\ 50.1.0.dmg
com.apple.diskimages.fsck:
00000000  D4 38 AA 6D 1A CC C0 27 CE 2A DB 77 F6 51 5D 80  |.8.m...'.*.w.Q].|
00000010  96 BC 72 4F                                      |..rO|
00000014
com.apple.diskimages.recentcksum:
00000000  69 3A 34 33 35 30 34 39 37 20 6F 6E 20 30 34 42  |i:4350497 on 04B|
00000010  33 39 44 36 37 2D 45 35 36 37 2D 33 45 45 34 2D  |39D67-E567-3EE4-|
00000020  42 34 38 45 2D 38 44 43 37 42 43 46 36 44 34 38  |B48E-8DC7BCF6D48|
00000030  46 20 40 20 31 34 38 32 35 39 39 33 39 30 20 2D  |F @ 1482599390 -|
00000040  20 43 52 43 33 32 3A 24 41 43 42 34 36 44 34 31  | CRC32:$ACB46D41|
00000050
com.apple.metadata:kMDItemDownloadedDate:
00000000  62 70 6C 69 73 74 30 30 A1 01 33 41 BE 0E E3 62  |bplist00..3A...b|
00000010  AC 63 5A 08 0A 00 00 00 00 00 00 01 01 00 00 00  |.cZ.............|
00000020  00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 13                                   |.....|
00000035
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 65 68 74  |bplist00..._.eht|
00000010  74 70 73 3A 2F 2F 64 6F 77 6E 6C 6F 61 64 2D 69  |tps://download-i|
00000020  6E 73 74 61 6C 6C 65 72 2E 63 64 6E 2E 6D 6F 7A  |nstaller.cdn.moz|
00000030  69 6C 6C 61 2E 6E 65 74 2F 70 75 62 2F 66 69 72  |illa.net/pub/fir|
00000040  65 66 6F 78 2F 72 65 6C 65 61 73 65 73 2F 35 30  |efox/releases/50|
00000050  2E 31 2E 30 2F 6D 61 63 2F 65 6E 2D 55 53 2F 46  |.1.0/mac/en-US/F|
00000060  69 72 65 66 6F 78 25 32 30 35 30 2E 31 2E 30 2E  |irefox%2050.1.0.|
00000070  64 6D 67 5F 10 32 68 74 74 70 73 3A 2F 2F 77 77  |dmg_.2https://ww|
00000080  77 2E 6D 6F 7A 69 6C 6C 61 2E 6F 72 67 2F 65 6E  |w.mozilla.org/en|
00000090  2D 55 53 2F 66 69 72 65 66 6F 78 2F 6E 65 77 2F  |-US/firefox/new/|
000000A0  3F 73 63 65 6E 65 3D 32 08 0B 73 00 00 00 00 00  |?scene=2..s.....|
000000B0  00 01 01 00 00 00 00 00 00 00 03 00 00 00 00 00  |................|
000000C0  00 00 00 00 00 00 00 00 00 A8                    |..........|
000000cb
com.apple.quarantine:
00000000  30 30 38 33 3B 35 38 35 65 61 62 64 65 3B 53 61  |0083;585eabde;Sa|
00000010  66 61 72 69 3B 36 39 35 41 44 44 35 37 2D 45 37  |fari;695ADD57-E7|
00000020  32 32 2D 34 46 34 45 2D 38 46 39 39 2D 35 45 46  |22-4F4E-8F99-5EF|
00000030  37 41 38 44 46 39 32 39 33                       |7A8DF9293|
00000039
```



On some newer systems, when an email attachment has been downloaded, a few extended attributes get attached to that file.

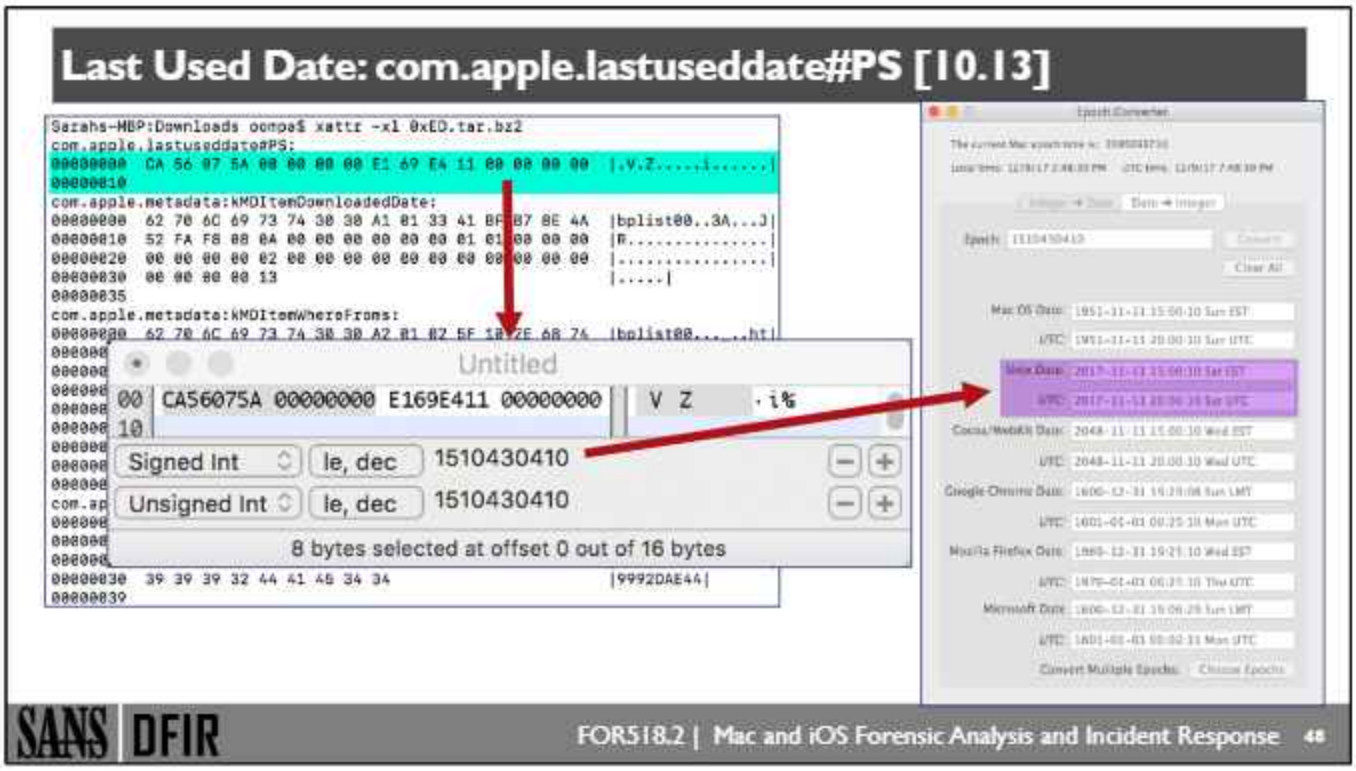
- `com.apple.metadata:com_apple_mail_dateReceived`: Binary plist containing the timestamp of when the email message was received
- `com.apple.metadata:com_apple_mail_dateSent`: Binary plist containing the timestamp of when the email message was sent
- `com.apple.metadata:com_apple_mail_isRemoteAttachment`: Binary plist containing a binary value if the attachment is local (0) or remote (1)

Another extended attribute, "`com.apple.metadata:kMDLabel_sztvaisjvgoqydo5khybt7gk6a`", contains unknown, possibly encrypted data; however, it does appear to get written on downloaded email attachments and other files sent/received. The string at the end of the attribute name appears to be unique to specific hardware.

Note the application found in the "`com.apple.quarantine`" attribute. It shows the file was downloaded using "Mail". The hex timestamp shows when the file was downloaded, as it does with other files.

```
bit:Downloads oompa$ xattr -xl folioxmlg5new93448788.pdf
com.apple.metadata:com_apple_mail_dateReceived:
00000000  62 70 6C 69 73 74 30 30 33 41 BE 09 9C EE 00 00  |bplist003A......|
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 09 9C E7 00 00  |bplist003A......|
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 36 22  |bplist00...._.6"|
00000010  47 72 61 6E 64 20 48 79 61 74 74 20 57 61 73 68  |Grand Hyatt Wash|
00000020  69 6E 67 74 6F 6E 22 3C 4E 41 2E 43 75 73 74 6F  |ington"<NA.Custo|
00000030  6D 65 72 53 65 72 76 69 63 65 40 48 79 61 74 74  |merService@Hyatt|
00000040  2E 63 6F 6D 3E 5F 10 21 59 6F 75 72 20 47 72 61  |.com>_.!Your Gra|
00000050  6E 64 20 48 79 61 74 74 20 57 61 73 68 69 6E 67  |nd Hyatt Washing|
00000060  74 6F 6E 20 65 42 69 6C 6C 5F 10 45 6D 65 73 73  |ton eBill_.Emess|
00000070  61 67 65 3A 25 33 43 32 30 31 36 31 32 32 30 31  |age:%3C201612201|
00000080  37 30 37 2E 75 42 4B 48 37 70 38 63 30 31 39 38  |707.uBKH7p8c0198|
00000090  36 31 40 69 61 70 69 6E 66 72 73 6D 74 70 30 31  |61@iapinfrsmtp01|
000000A0  2E 72 73 73 2E 68 79 61 74 74 2E 63 6F 6D 25 33  |.rss.hyatt.com%3|
000000B0  45 08 0C 45 69 00 00 00 00 00 00 01 01 00 00 00  |E..Ei...........|
000000C0  00 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00  |................|
000000D0  00 00 00 00 B1                                   |.....|
000000d5
```

```
com.apple.metadata:kMDLabel_sztvaisjvgoqydo5khybt7gk6a:
00000000  F2 7E 90 64 8B 0F FD 96 51 39 E7 00 EA 4E EE 0B  |.~.d....Q9...N..|
00000010  9F 53 7A 1E C8 96 77 73 2B 14 25 AF B5 0C 16 4F  |.Sz...ws+.%....O|
[redacted]
000002C0  6F C1 1C E8 F0 81 11 9C 7F 6D 51 03 CD 4C 61 F0  |o........mQ..La.|
000002D0  66 96 18 FC D0 51 B3 71 C6                       |f....Q.q.|
000002d9
com.apple.quarantine:
00000000  30 30 38 32 3B 35 38 35 65 61 64 65 39 3B 4D 61  |0082;585eade9;Ma|
00000010  69 6C 3B                                         |il;|
00000013
```



New extended attributes show up all the time. This one is particularly useful to forensic analysts. This one will keep a timestamp of when a file was last used, as it pertains to the file system.

This will generally be seen when the files have been used in the Finder windows; however, if a file is opened using the "open" command in the terminal, the timestamp will also be updated.

Not all file types will have this attribute, and research continues to determine which ones do and do not.

AirDrop Artifacts

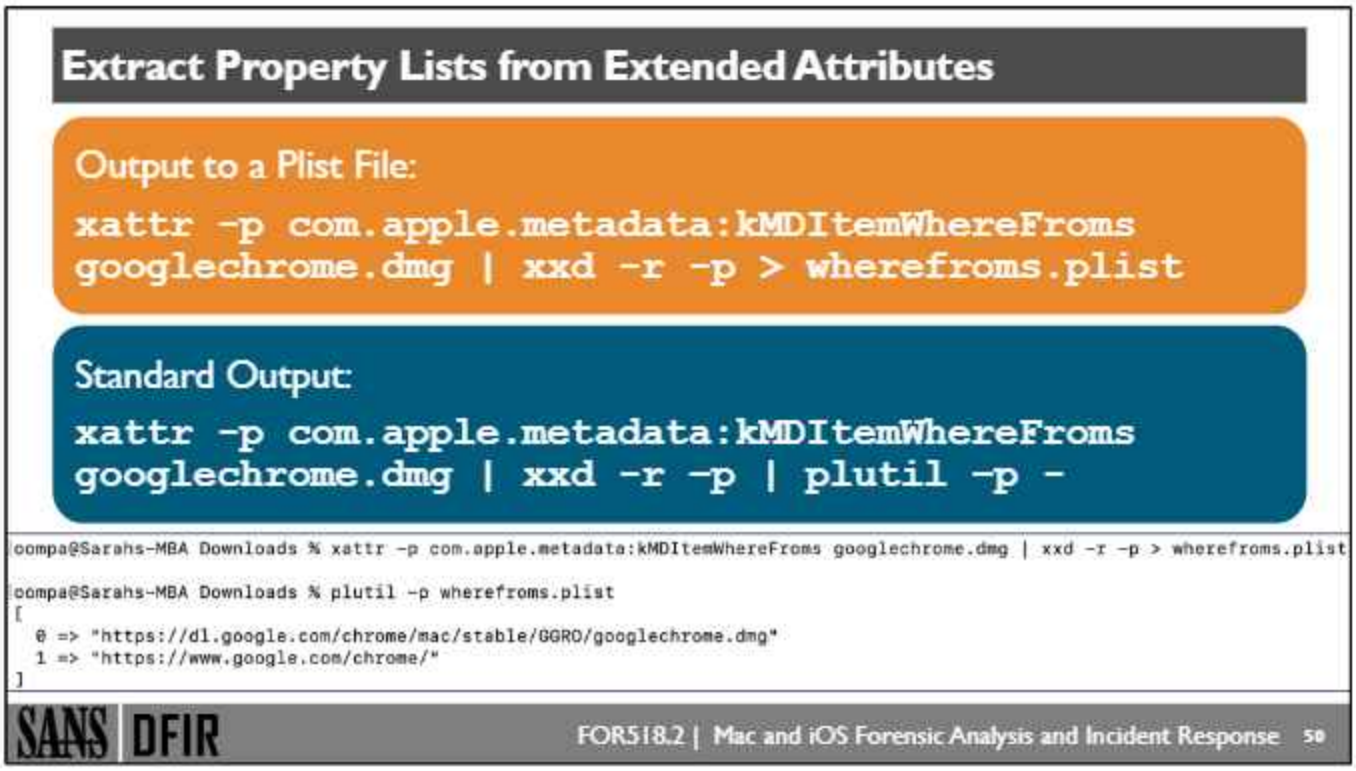
```
oompa@MBP-M1 Downloads % xattr -xl IMG_0005.heic
com.apple.cscachefs:
00000000  53 63 A7 59 DB BD D9 FF 5B 96 B7 58 1A C7 56 54  |Sc.Y....[..X..VT|
00000010  5F 38 AD DA 04 7D 00 00 00 CA 18 D3 62 00 00 00  |_8...}.....b...|
00000020  00 35 17 51 00 00 00 00 00 69 45 1E 00 00 00 00  |.5.Q.....iE.....|
00000030  00 01 3C 3F DE 25 F2 6D 3E F9 38 27 51 19 5D 84  |..<?.%.m>.8'Q.].|
00000040  F0 04 2F DA 3E EC                                |../.>.|
00000046
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5C 45 6C 77 6F  |bplist00...\Elwo|
00000010  6F 64 20 42 6C 75 65 73 6F 10 0F 00 45 00 6C 00  |od Bluesо.....|
00000020  77 00 6F 00 6F 00 64 20 19 00 73 00 20 00 69 00  |w.....d ..s. .i.|
00000030  50 00 68 00 6F 00 6E 00 65 00 0B 16 00 00 00 00  |P.....e.....|
00000040  00 00 01 01 00 00 00 00 00 00 00 03 00 00 00 00  |................|
00000050  00 00 00 00 00 00 00 00 00 00 00 39              |...........|
0000005C
com.apple.quarantine:
00000000  30 30 38 31 3B 36 32 64 33 31 39 30 3B 73 68  |0081;62d31900;sh|
00000010  61 72 69 6E 67 64 3B 46 35 43 38 43 43 44 46 2D  |aringd;F5C8CCDF-|
00000020  37 44 35 37 2D 34 33 46 41 2D 41 35 42 35 2D 43  |7D57-43FA-A5B5-C|
00000030  35 45 38 33 35 30 41 34 37 45 36                 |5E8350A47E6|
00000038
```

AirDrop is available on newer 10.7+ systems. This application allows users to "drop" files to other users in their vicinity. Note: This does not necessarily mean the user has to be on the same network it works using Continuity technologies over Wi-Fi and Bluetooth.

In the example, Elwood Blues wants to send a photo. If this file is saved, it will by default be saved to the `~/Downloads` directory.

Extended attributes for the file on the recipient's system show the file was from 'Elwood Blues' as this is the name associated with the device. This may also show hostname and device information. The name can be changed by the user, so it is not a foolproof way of determining whom a file came from. For Mac devices this is usually the hostname of the device.

```
word:Downloads oompa$ xattr -xl IMG_1626.JPG
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 56 69 50 68 6F  |bplist00...ViPho|
00000010  6E 65 58 6D 69 50 68 6F 6E 65 36 08 0B 12 00 00  |neXmiPhone6.....|
00000020  00 00 00 00 01 01 00 00 00 00 00 00 00 03 00 00  |................|
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 1B        |..............|
0000003e
```



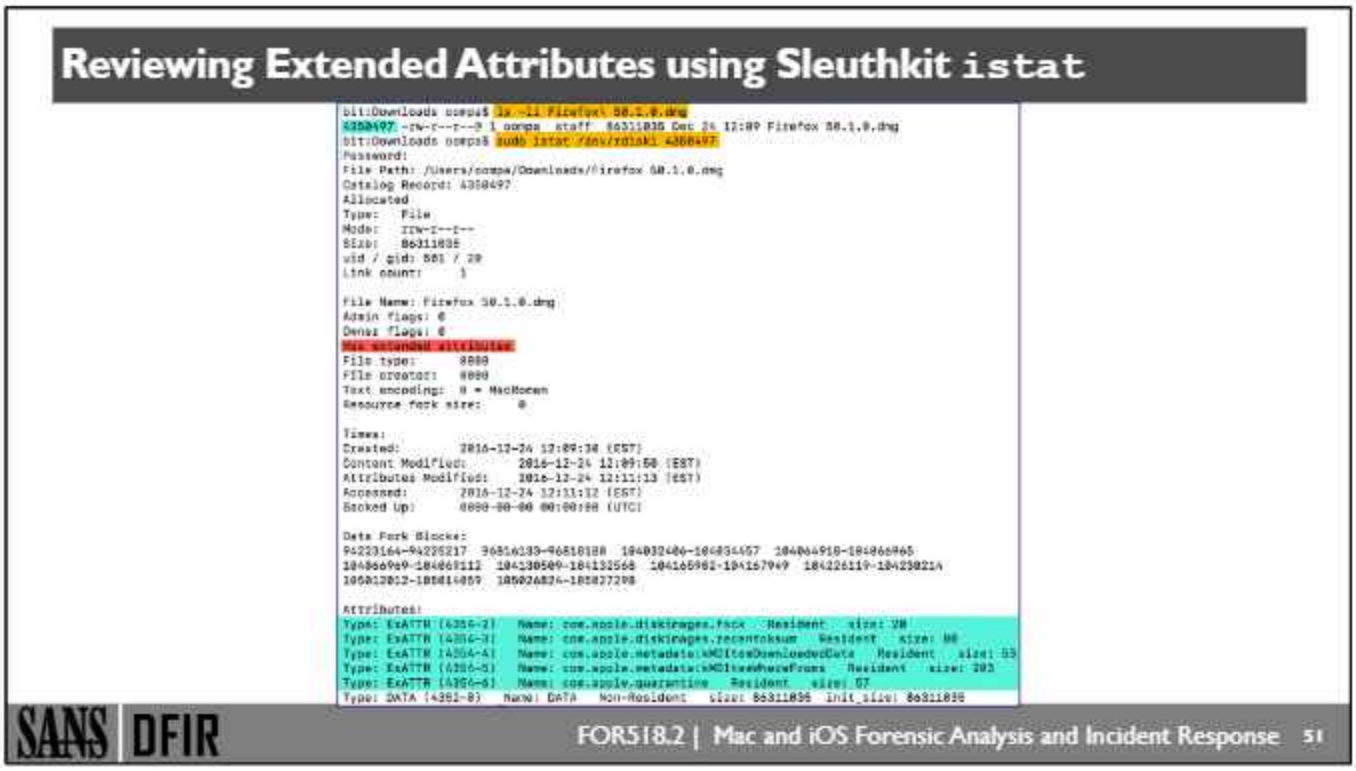
Some extended attributes contain binary property lists. To extract these for more detailed analysis, we can use the `xattr` command with the `-p` option. This option "prints" the attribute data.

Using only the `-p` option, it will print the hex version of the data. To create a file, we must use the `xxd` command to "revert to binary" (`-r`) and print to "plaintext" (`-p`). We can use the ">" symbol to redirect the output to a file.

Together, the commands will look like this. This command prints the output of the `com.apple.metadata:kMDItemWhereFroms` attribute to a file named `wherefroms.plist`.

```
xattr -p com.apple.metadata:kMDItemWhereFroms googlechrome.dmg | xxd -r -p >
wherefroms.plist
```

To print a binary plist directly to standard out on the Terminal, use the following command format:
`xattr -p <attribute name> <file> | xxd -r -p | plutil -p -`



The first command shown in the left screenshot, "`ls -li`", shows the file's inode (CNID) number that we can use with the next command.

The Sleuthkit's (TSK) `istat` command prints the names and sizes of extended attributes in its "`Attributes`" section, shown in the screenshot above for the file with CNID (inode) 4350497.

Each attribute contains a TSK-assigned attribute number. For extended attributes, this number is `4354-#`, where # is filled in for each individual attribute. This can be used to extract the contents of the attribute using Sleuthkit's `icat` utility, shown later.

```
bit:Downloads oompa$ ls -li Firefox\ 50.1.0.dmg
4350497 -rw-r--r--@ 1 oompa  staff  86311035 Dec 24 12:09 Firefox 50.1.0.dmg
bit:Downloads oompa$ sudo istat /dev/rdisk1 4350497
Password:
File Path: /Users/oompa/Downloads/Firefox 50.1.0.dmg
Catalog Record: 4350497
Allocated
Type:    File
Mode:    rrw-r--r--
Size:    86311035
uid / gid: 501 / 20
Link count:     1

File Name: Firefox 50.1.0.dmg
Admin flags: 0
Owner flags: 0
Has extended attributes
File type:      0000
File creator:   0000
Text encoding:  0 = MacRoman
Resource fork size:     0

Times:
Created:        2016-12-24 12:09:30 (EST)
Content Modified:       2016-12-24 12:09:50 (EST)
Attributes Modified:    2016-12-24 12:11:13 (EST)
Accessed:       2016-12-24 12:11:12 (EST)
Backed Up:      0000-00-00 00:00:00 (UTC)

Data Fork Blocks:
94223164-94225217  96816133-96818180  104032406-104034457  104064918-104066965
104066969-104069112  104130509-104132568  104165902-104167949  104226119-104230214
105012012-105014059  105026824-105027298

Attributes:
Type: ExATTR (4354-2)   Name: com.apple.diskimages.fsck   Resident   size: 20
Type: ExATTR (4354-3)   Name: com.apple.diskimages.recentcksum   Resident   size: 80
Type: ExATTR (4354-4)   Name: com.apple.metadata:kMDItemDownloadedDate   Resident   size: 53
Type: ExATTR (4354-5)   Name: com.apple.metadata:kMDItemWhereFroms   Resident   size: 203
Type: ExATTR (4354-6)   Name: com.apple.quarantine   Resident   size: 57
Type: DATA (4352-0)   Name: DATA   Non-Resident   size: 86311035  init_size: 86311035
```

Extracting Extended Attributes with Sleuthkit `icat`

Local Disk

```
bit:Downloads oompa$ sudo icat /dev/rdisk1 4350497-4354-5 | plutil -p -
[
  0 => "https://download-installer.cdn.mozilla.net/pub/firefox/releases/50.1.0/mac/en-US/Firefox%2050.1.0.dmg"
  1 => "https://www.mozilla.org/en-US/firefox/new/?scene=2"
]
```

E01 Image

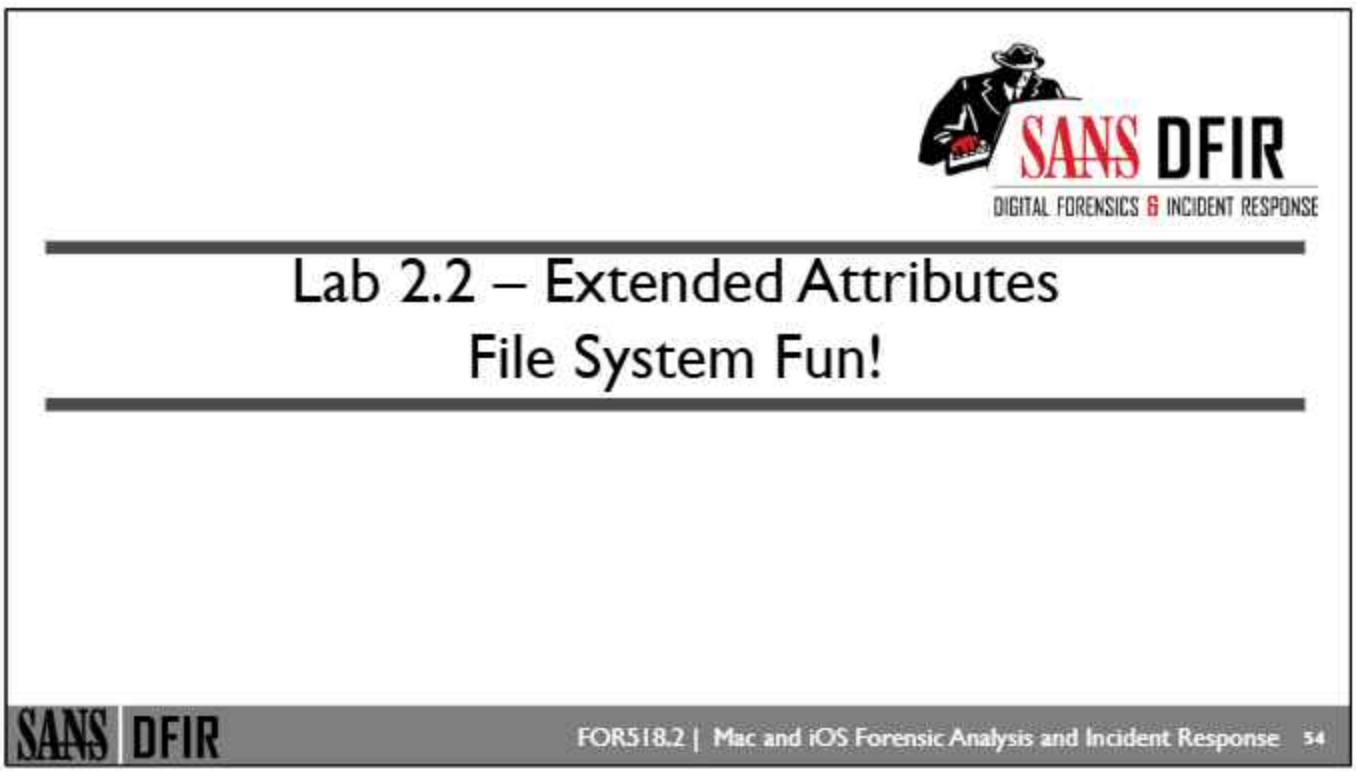
```
bit:FOR518 Images oompa$ icat dademurphy.E01 380562-4354-3 | plutil -p -
[
  0 => "http://download-installer.cdn.mozilla.net/pub/firefox/releases/26.0/mac/en-US/Firefox%2026.0.dmg"
  1 => "http://www.mozilla.org/en-US/firefox/new/"
]
```

SANS DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 53

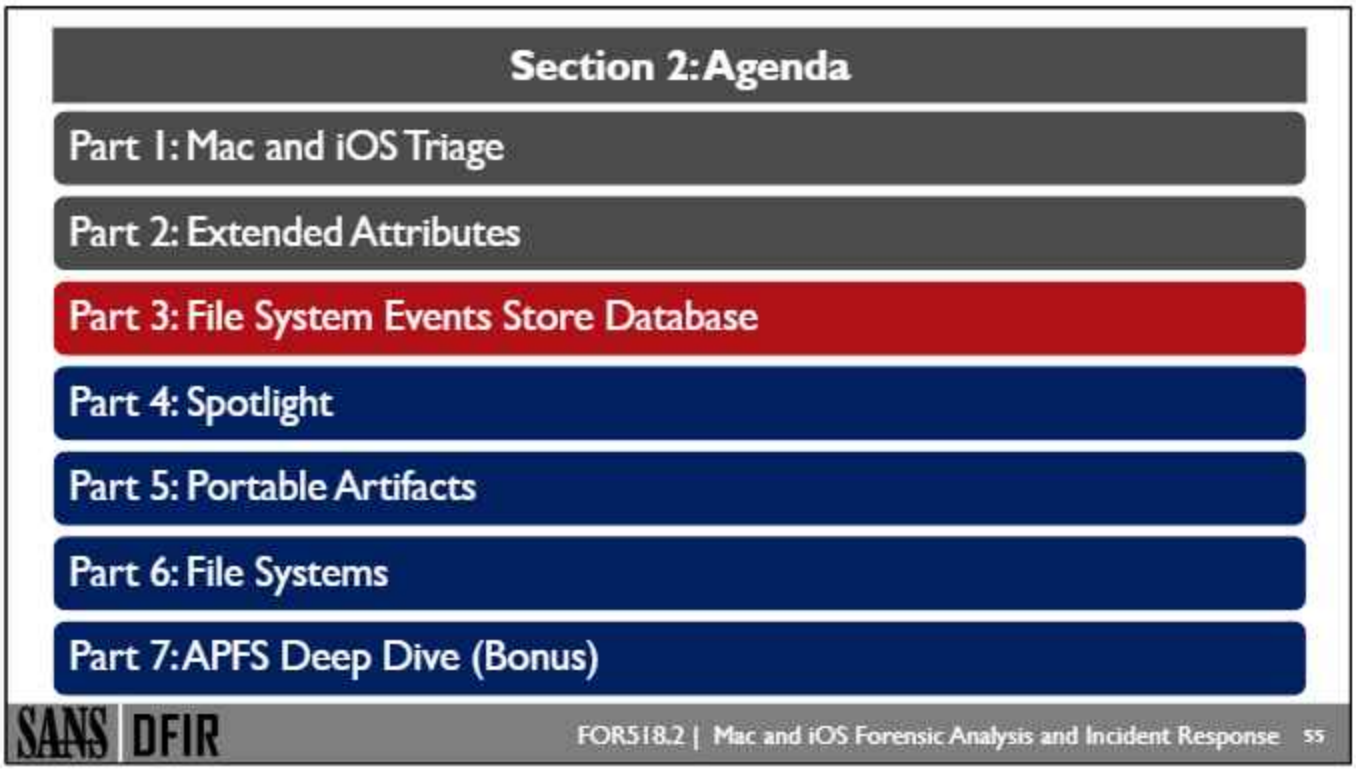
The Sleuthkit `icat` command can be used to print the data contained in a specific attribute. The TSK attribute number is appended to the CNID (inode) of the file to print the specific attribute.

The top screenshot shows an example of this using the local disk (`/dev/rdisk1`). The bottom screenshot is an example using an E01 image.

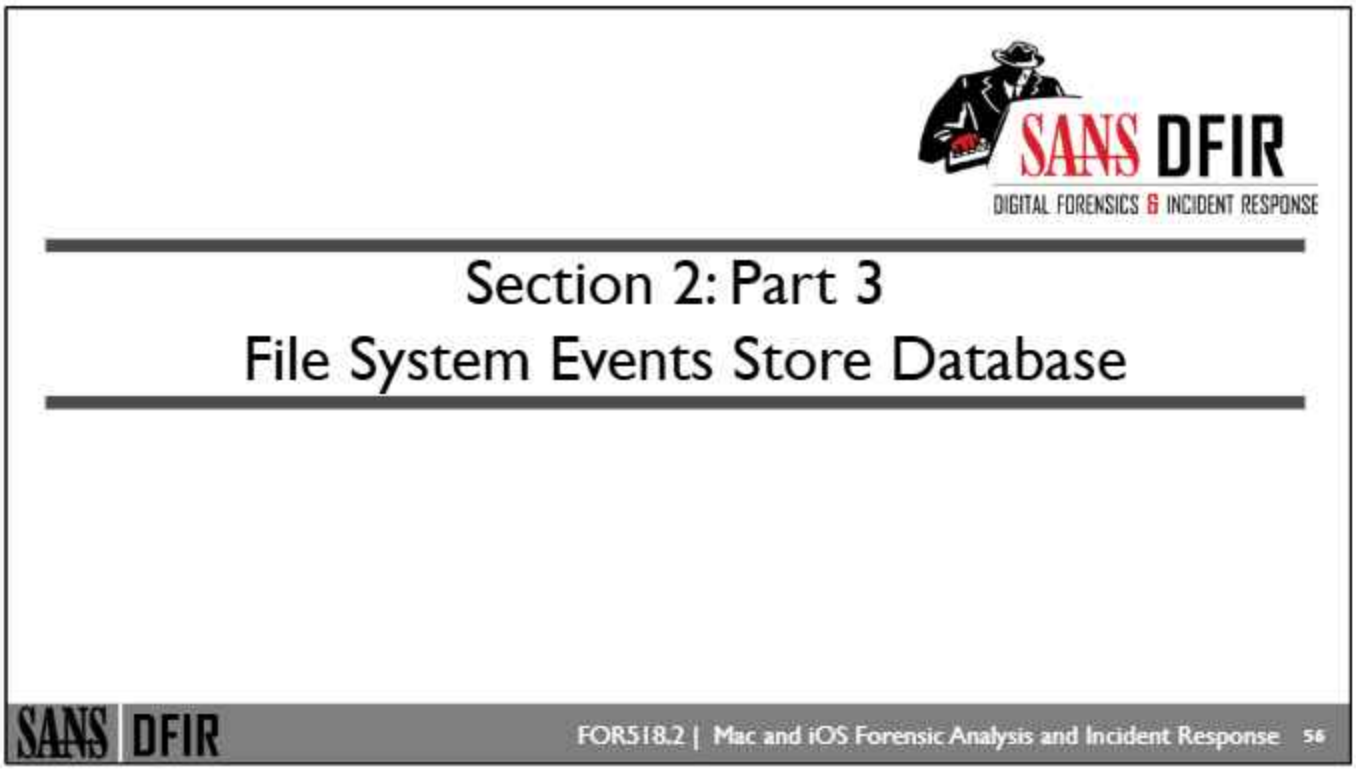
The screenshots show the `icat` command printing the extended attributes identified by `4354-5` for CNID `4350497` and `4354-3` for the CNID `380562`. This is the `com.apple.metadata:kMDItemWhereFroms` attribute for each file.



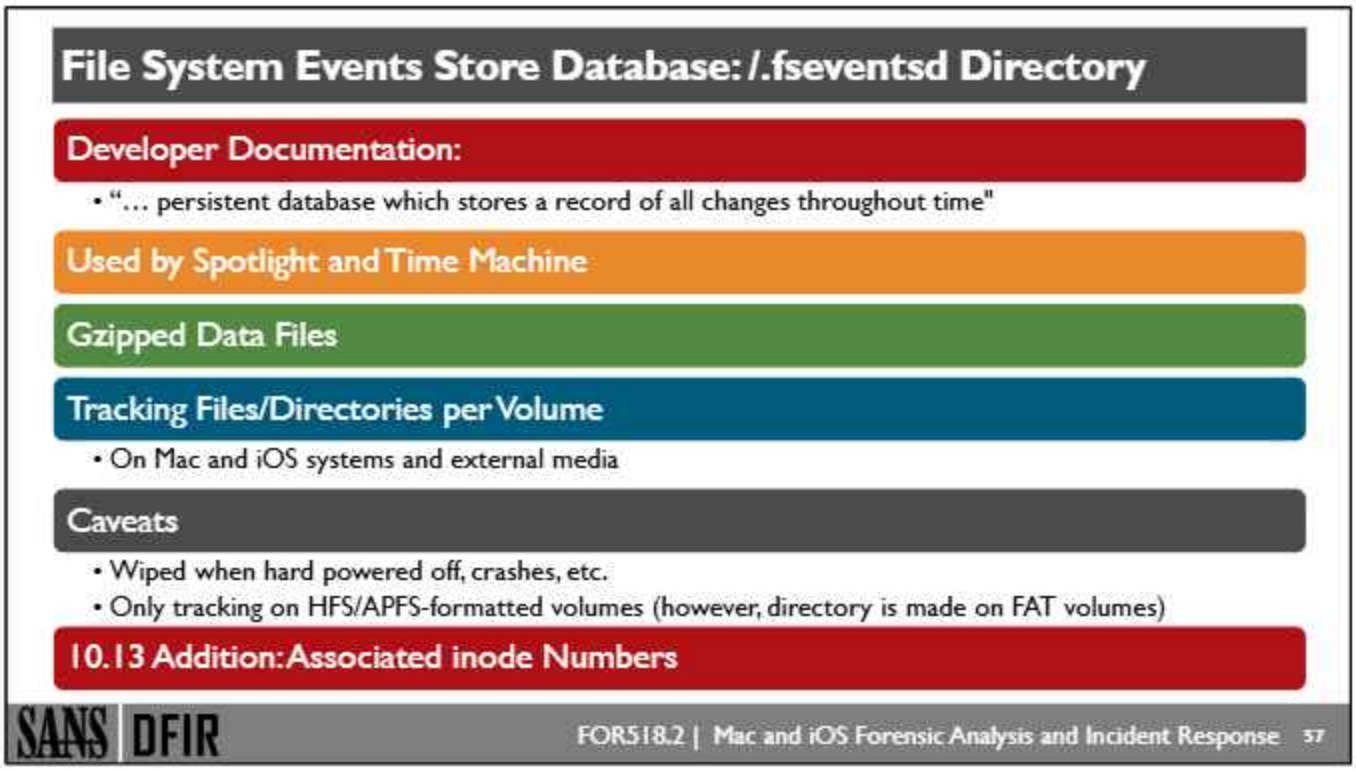
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.



Each volume connected to a Mac system will have a `/.fseventsd` directory created. This directory is the File System Events Store Database that is responsible for storing file system changes on the volume.

Spotlight and Time Machine use this database to determine what files are new or have changed metadata properties. This directory contains gzipped files that require root privileges to unzip and view them. While this artifact can be incredibly useful, it does have some caveats. It can get wiped out when the system gets hard powered off or during a crash. However, if file carving is a possibility, these files can be found by searching for gzip files.

The `/.fseventsd` directory contains gzipped files, each named incrementally with no file extension. The `fseventsd-uuid` file contains the GUID of the FSEvents database. On an HDD volume, this should stay persistent, barring system malfunction. On an external USB drive, it is likely to change each time the drive is inserted into a system. FSEvent UUIDs will show changes in the system logs if you do a search for "`fsevents`". The /.fseventsd directory will be created on non-HFS+ volumes; however, it does not appear to create the database files.

The format of the unzipped files is noted to have changed over time in the Apple Developer Documentation and has not been thoroughly documented; however, filenames and file paths should be human readable. Therefore, we can run the strings utility on these files to find remnants of deleted directories or files. The results are shown on the next slide.

References:
Apple Developer Documentation: File System Events Programming Guide
https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005289-CH1-SW1
"FSEvents and other interesting files" by Derrick Donnelly of BlackBag Technologies, Enfuse 2016
Apple Developer Documentation: FSEvents Reference
https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html

```
sh-3.2# pwd
/.fseventsd
sh-3.2# ls -lAr
total 74168
-rw-------  1 root  admin       36 Jun 30 17:09 fseventsd-uuid
-rw-------  1 root  admin    13488 Jul  6 10:33 0000000011ea5d4a
-rw-------  1 root  admin     3986 Jul  6 09:48 0000000011e942cb
-rw-------  1 root  admin    16438 Jul  6 09:39 0000000011e85439
-rw-------  1 root  admin    18365 Jul  6 08:38 00000000113fc77c
-rw-------  1 root  admin    17952 Jul  6 07:13 0000000011344ed6
-rw-------  1 root  admin     8247 Jul  6 04:07 000000001131f4eb
-rw-------  1 root  admin    12524 Jul  5 22:32 00000000112fcad6
-rw-------  1 root  admin    10181 Jul  5 20:50 0000000011267ce0
-rw-------  1 root  admin    20699 Jul  5 20:14 0000000011257dc1
-rw-------  1 root  admin    15236 Jul  5 20:13 0000000011242ded
```

```
sh-3.2# file *
00000000003fa974:    gzip compressed data, from Unix
0000000000417ee4:    gzip compressed data, from Unix
000000000042a918:    gzip compressed data, from Unix
0000000000450189:    gzip compressed data, from Unix
00000000004644c5:    gzip compressed data, from Unix
000000000046fa05:    gzip compressed data, from Unix
000000000047af8f:    gzip compressed data, from Unix
```

```
nibble:extracted sledwards$ xxd 00000000125482b4
0000000: 3153 4c44 0727 dc56 dc00 0000 0091 5554  1SLD.'.V......UT
0000010: 1200 0000 0000 0000 022e 4453 5f53 746f  ..........DS_Sto
0000020: 7265 00ca 8154 1200 0000 0055 0080 002e  re...T.....U....
0000030: 5472 6173 6865 7300 9354 5412 0000 0000  Trashes..TT.....
0000040: 4800 0001 476f 6f67 6c65 4368 726f 6d65  H...GoogleChrome
0000050: 5374 616e 6461 6c6f 6e65 456e 7465 7270  StandaloneEnterp
0000060: 7269 7365 2028 3129 2e6d 7369 009b 8254  rise (1).msi...T
0000070: 1200 0000 0055 0780 004b 656c 6968 6f73  .....U...Kelihos
0000080: 2d48 6c75 782d 3230 3133 2e7a 6970 00b3  -Hlux-2013.zip..
0000090: 8254 1200 0000 0055 0780 0062 6c61 6832  .T.....U...blah2
00000a0: 2e74 7874 00f4 5e54 1200 0000 0008 0080  .txt..^T........
00000b0: 0062 6c61 6833 2e74 7874 00d3 7254 1200  .blah3.txt..rT..
00000c0: 0000 0011 0180 0074 6573 742e 7478 7400  .......test.txt.
00000d0: 5c5e 5412 0000 0000 1500 8000            \^T.........
```

FSEvents Parser: Free Python Script by Nicole Ibrahim https://github.com/dlcowen/FSEventsParser

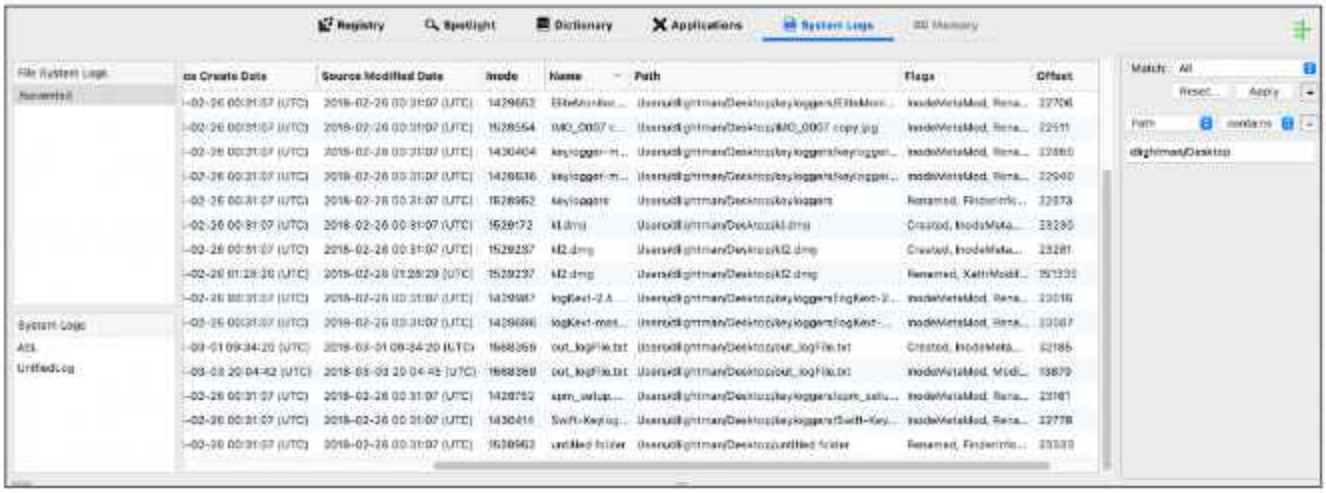
Python: SQLite Database or CSV Output

Table: fsevents

	record_filename	record_mask	record_mask_hex	name_date_stamp	record_id
	Filter	Filter	Filter	Filter	Filter
4	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/1DD5913F-786B-42DB-AFCB-6B84FF8320FA.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10459185
5	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/2516E71C-C9FC-43EF-AF38-D73A78E5F631.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10456685
6	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/2631C8E1-8EE9-48C4-86DB-878EDC829B02.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10474445
7	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/3C1DDA82-DE36-451A-A211-72CED8368AF6.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10489409
8	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/480F86C6-675F-4896-A8F9-2D9F05503358.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10461231
9	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/526F8687-7F6B-4357-B719-64C505E35350.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10467913
10	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/5664644-A255-4DA4-B4BE-85C7E16529C0.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10476878
11	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/9BAB03AD-1018-4BAC-86A6-DE44331C0C3E.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10472117
12	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/CF1D09A5-CF98-4CD5-9788-12E8BE389734.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10466258
13	Users/oompa/Library/Calendars/4BCC813B-AD04-4021-A206-18D749504FBE.calendar/Events/D2E48483-9DC8-4F99-AEF2-C8D9F640E7C0.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10475257
14	Users/oompa/Library/Calendars/734F76E5-5012-4812-A372-95C817D15861.calendar/Events/717A3166-CF95-41F1-98E3-888E0E35154.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10463051
15	Users/oompa/Library/Calendars/734F76E5-5012-4812-A372-95C817D15861.calendar/Events/77EA158F-2814-4BF4-969F-FE821982C8F5.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10488503

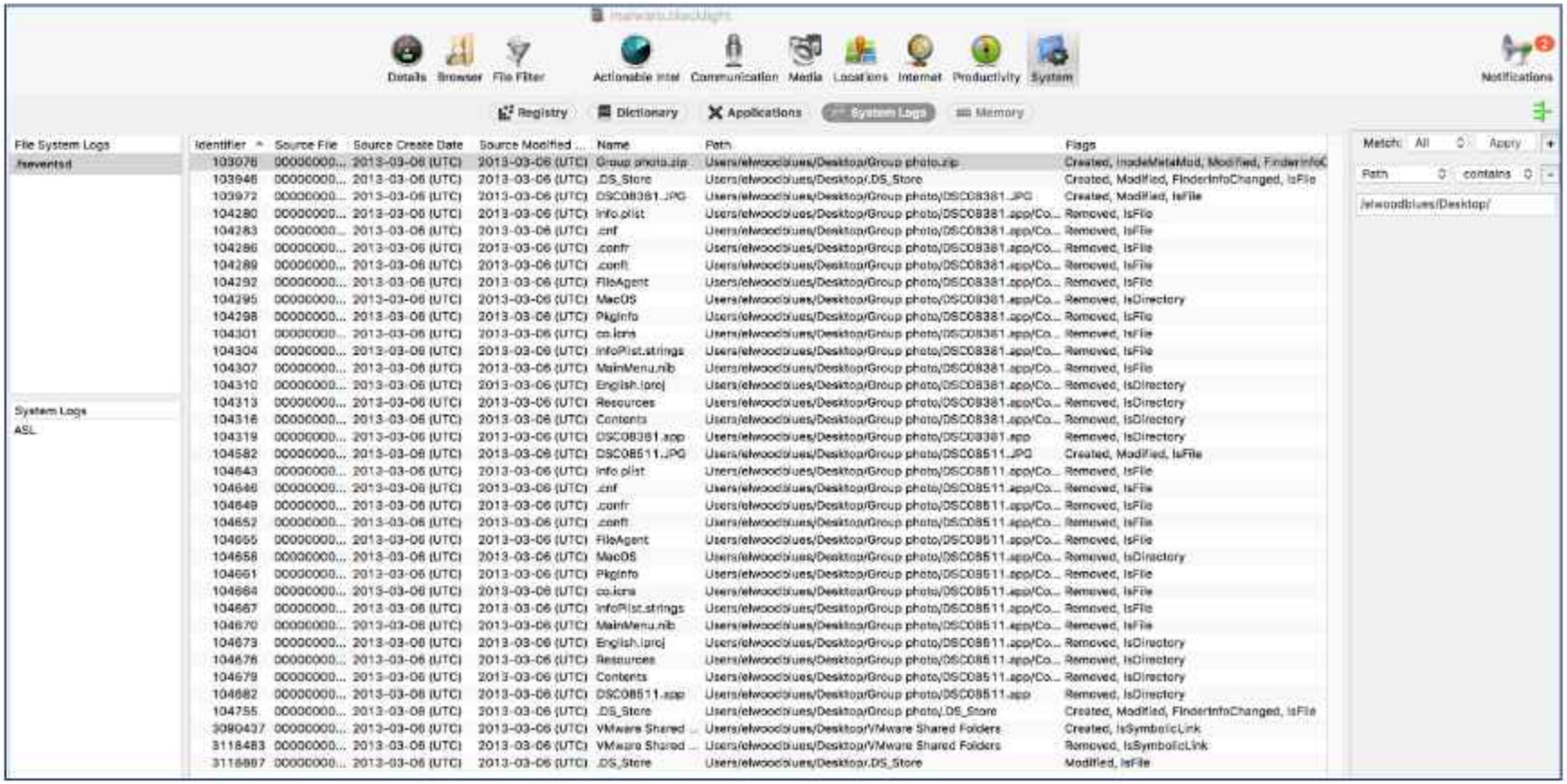
A Python script has been developed that allows forensic analysis of this data in a more human-readable format. This script takes in a directory of fsevent files and will output a SQLite database and CSV of parsed data. The data can then be analyzed in the analyst's preferred viewer.

FSEvents Parser: Inspector



In Inspector, a file system events parser is built in using the advanced processing capabilities. This can be run when the image is added or later in the "Evidence Status" processor on the left sidebar.

Once run, the analyst will find the items in the System | System Logs area of Inspector, as shown in the example above. On the right side, there is a filter icon that can be used to filter the results for whatever the analyst might be looking for. This is highly recommended, as results for file system events tend to be quite verbose.



FSEvents Analysis: Example Scenario [1]

- Scenario: Newly created USB HFS+ volume
- Created directories and added files to mounted USB volume

	wd	mask_hex	filename	mask
1	18158642773867881568	0x00000002	NULL	Mount;
2	18158642773867883385	0x08000001	Directory_1	FolderEvent;Renamed;
3	18158642773867883520	0x08000001	Directory_2	FolderEvent;Renamed;
4	18158642773867883566	0x08000001	untitled folder	FolderEvent;Renamed;FolderCreated;
5	18158642773867883587	0x08000001	Directory_3	FolderEvent;Renamed;
6	18158642773867885468	0x65078000	Directory_1/DropboxInstaller.dmg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
7	18158642773867885528	0x65038000	Directory_1/ExifTool-10.36.dmg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;FileEvent;
8	18158642773867886607	0x65078000	Directory_1/HexRand.zip	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
9	18158642773867886915	0x65078000	Directory_2/IMG_2311.JPG.jpeg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
10	18158642773867886936	0x65078000	Directory_2/IMG_2326.JPG	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
11	18158642773867887932	0x65008000	.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;FileEvent;
12	18158642773867887968	0x08000001	Directory_3/untitled folder	FolderEvent;Renamed;FolderCreated;
13	18158642773867887986	0x08000001	Directory_3/Directory_4	FolderEvent;Renamed;

Using the free Python script to parse the file system events, they can be viewed and queried in an SQLite database, as shown for the next few examples.

The scenario used for the following examples shows a new clean USB thumb drive that was wiped and reformatted with HFS+. The volume was created and mounted, and three directories were added to it: `Directory_1`, `Directory_2`, and `Directory_3`. The file system events show a "`Mount`" activity with three "`FolderEvent;Renamed`" activities for each of the three directories. This is because the first directory created was initially called "`untitled folder`", as shown above, which just got renamed.

The next section shows files being added to various directories: Installer DMG/ZIP files to `Directory_1`, images to `Directory_2`, and a new folder created in `Directory_3`, named `Directory_4`. These have the "`Created`" activity associated with each one.

Note the `.DS_Store` is shown. This means the user added these files using the Finder GUI window.

Reference:
https://github.com/dlcowen/FSEventsParser

	wd	mask_hex	filename	mask
1	18158642773887881568	0x00000002	NULL	Mount;
2	18158642773887883395	0x08000001	Directory_1	FolderEvent;Renamed;
3	18158642773887883520	0x08000001	Directory_2	FolderEvent;Renamed;
4	18158642773887883566	0x88000001	untitled folder	FolderEvent;Renamed;FolderCreated;
5	18158642773887883567	0x08000001	Directory_3	FolderEvent;Renamed;
6	18158642773887885468	0x55078000	Directory_1/DropboxInstaller.dmg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
7	18158642773887885528	0x55038000	Directory_1/ExifTool-10.36.dmg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;FileEvent;
8	18158642773887885607	0x55078000	Directory_1/HexFiend.zip	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
9	18158642773887886915	0x55078000	Directory_2/IMG_2311.JPG.jpeg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
10	18158642773887886936	0x55078000	Directory_2/IMG_2326.JPG	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
11	18158642773887887932	0x55008000	.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;FileEvent;
12	18158642773887887966	0x88000001	Directory_3/untitled folder	FolderEvent;Renamed;FolderCreated;
13	18158642773887887969	0x08000001	Directory_3/Directory_4	FolderEvent;Renamed;

FSEvents Analysis: Example Scenario [2]

Created a new text document with TextEdit (Document Versions)

Next, the user opened the TextEdit application and created an RTF document on the USB thumb drive. The initial name of the document was "`Untitled 9.rtf`". Once the user saved and renamed the document, it was saved as "`testdoc.rtf`". You can see temporary files being created before the document is actually saved as "`testdoc.rtf`".

Note all the "`.DocumentRevisions`" and "`.TemporaryItems`" file paths. This is how macOS Document Versions works; the details will be revealed in a later module.

Reference:
https://github.com/dlcowen/FSEventsParser

	filename	mask
	Filter	Filter
15	Directory_3/Directory_4/Untitled 9.rtf.sb-3b0a3d1b-CoR8Fc/Untitled 9.rtf	Modified;Renamed;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;FileEvent;
16	Directory_3/Directory_4/Untitled 9.rtf.sb-3b0a3d1b-CoR8Fc	FolderEvent;Removed;FolderCreated;
17	.DocumentRevisions-V100	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
18	.DocumentRevisions-V100/db-V1	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
19	.DocumentRevisions-V100/purgatory	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
20	.DocumentRevisions-V100/.cs	FolderEvent;FolderCreated;
21	.DocumentRevisions-V100/.cs/ChunkStoreDatabase	Created;FileEvent;
22	.DocumentRevisions-V100/.cs/ChunkStoreDatabase-journal	Modified;InodeMetaMod;Created;Removed;FileEvent;
23	.DocumentRevisions-V100/.cs/ChunkStoreDatabase-wal	InodeMetaMod;Created;FileEvent;
24	.DocumentRevisions-V100/db-V1/db.sqlite	Created;FileEvent;
25	.DocumentRevisions-V100/db-V1/db.sqlite-journal	Modified;InodeMetaMod;Created;Removed;FileEvent;
26	.DocumentRevisions-V100/db-V1/db.sqlite-wal	InodeMetaMod;Created;FileEvent;
27	.TemporaryItems	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
28	.TemporaryItems/folders.0	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
29	.TemporaryItems/folders.0/TemporaryItems	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
30	.DocumentRevisions-V100/staging	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
31	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)/metadata	Modified;Renamed;Created;FileEvent;
32	.DocumentRevisions-V100/metadata	Renamed;PermissionChange;FileEvent;
33	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)/LibraryStatus	Modified;Renamed;Created;FileEvent;
34	.DocumentRevisions-V100/LibraryStatus	Renamed;FileEvent;
35	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)	FolderEvent;Removed;FolderCreated;
36	Directory_3/Directory_4/Untitled 9.rtf	Renamed;ExtendedAttrModified;FinderInfoMod;FileEvent;
37	Directory_3/Directory_4/textdoc.rtf	Renamed;FileEvent;

FSEvents Analysis: Example Scenario [3]

Unzipped HexFiend.zip into Directory_1

	mask_hex	filename	mask
77	0xc0010001	.TemporaryItems/folders.501/Cleanup At Startup	FolderEvent;PermissionChange;FinderInfoMod;FolderCreated;
78	0x68000001	Directory_1/.BAH.boX1A	FolderEvent;Renamed;FolderCreated;
79	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;
80	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;
81	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;

- ... <cut for brevity>

	mask_hex	filename	mask
212	0x8c030001	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app	FolderEvent;Renamed;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
213	0x0a020001	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj	FolderEvent;Renamed;ExtendedAttrModified;Removed;
214	0x55008000	Directory_1/.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;FileEvent;
215	0x84010001	.Trashes	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
216	0x84010001	.Trashes/501	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
217	0x68020001	Directory_1/Hex Fiend.app	FolderEvent;Renamed;ExtendedAttrModified;FolderCreated;
218	0x00028000	Directory_1/Hex Fiend.app/Contents/Library/LaunchServices/com.ridiculousfish.H...	ExtendedAttrModified;FileEvent;
219	0x00020001	Directory_1/Hex Fiend.app/Contents/Resources	FolderEvent;ExtendedAttrModified;

The user then unzipped the installer file `HexFiend.zip` into `Directory_1`. The unzipping process creates many temporary files in ".TemporaryItems/". (For brevity, many are not shown.)

Once unzipped, you can see the "`Hex Fiend.app`" directory structure being written to disk with all the files that make up an application directory/bundle. The "`.Trashes`" folder is also created in this activity.

Reference:
https://github.com/dlcowen/FSEventsParser

	mask_hex	filename	mask
212	0x8c030001	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app	FolderEvent;Renamed;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
213	0x0a020001	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj	FolderEvent;Renamed;ExtendedAttrModified;Removed;
214	0x55008000	Directory_1/.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;FileEvent;
215	0x84010001	.Trashes	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
216	0x84010001	.Trashes/501	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
217	0x88020001	Directory_1/Hex Fiend.app	FolderEvent;Renamed;ExtendedAttrModified;FolderCreated;
218	0x00028000	Directory_1/Hex Fiend.app/Contents/Library/LaunchServices/com.ridiculousfish.H...	ExtendedAttrModified;FileEvent;
219	0x00020001	Directory_1/Hex Fiend.app/Contents/Resources	FolderEvent;ExtendedAttrModified;

	mask_hex	filename	mask
77	0xc0010001	.TemporaryItems/folders.501/Cleanup At Startup	FolderEvent;PermissionChange;FinderInfoMod;FolderCreated;
78	0x88000001	Directory_1/.BAH.boX1A	FolderEvent;Renamed;FolderCreated;
79	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;
80	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;
81	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;

FSEvents Analysis: Example Scenario [4]

- ExifTool-10.36.dmg file moved directories
- File removal
  - HexFiend.zip sent to trash, then trash was emptied
    - "Removed" added to mask
  - IMG_2326.JPG sent to trash (not emptied)

	mask_hex	filename	mask
294	0x08008000	Directory_1/ExifTool-10.36.dmg	Renamed;FileEvent;
295	0x08008000	Directory_2/ExifTool-10.36.dmg	Renamed;FileEvent;
296	0x08008000	Directory_1/HexFiend.zip	Renamed;FileEvent;
297	0x14008000	Directory_1/.DS_Store	Modified;InodeMetaMod;FileEvent;
298	0x0a008000	.Trashes/501/HexFiend.zip	Renamed;Removed;FileEvent;
299	0x5d078000	IMG_2326.JPG	Modified;Renamed;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
300	0x08008000	.Trashes/501/IMG_2326.JPG	Renamed;FileEvent;
301	0x57008000	.Trashes/501/.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;Removed;FileEvent;
302	0x14008000	.DS_Store	Modified;InodeMetaMod;FileEvent;

Trashed, Trash Emptied

Just Trashed

SANS DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 68

Finally, the user wants to remove files from the USB thumb drive.

The first file, "`ExifTool-10.36.dmg`", was sent to the trash, and the trash was emptied explicitly by the user. Note the "`Removed`" event for this item.

Second, the "`IMG_2326.JPG`" file was sent to the trash, but the user did not empty it. The mask does not show the file was removed. It can still be recovered from the "`.Trashes/501/`" directory on the USB thumb drive.

Reference:
https://github.com/dlcowen/FSEventsParser

	mask_hex	filename	mask
294	0x08008000	Directory_1/ExifTool-10.36.dmg	Renamed;FileEvent;
295	0x08008000	Directory_2/ExifTool-10.36.dmg	Renamed;FileEvent;
296	0x08008000	Directory_1/HexFiend.zip	Renamed;FileEvent;
297	0x14008000	Directory_1/.DS_Store	Modified;InodeMetaMod;FileEvent;
298	0x0a008000	.Trashes/501/HexFiend.zip	Renamed;Removed;FileEvent;
299	0x5d078000	IMG_2326.JPG	Modified;Renamed;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
300	0x08008000	.Trashes/501/IMG_2326.JPG	Renamed;FileEvent;
301	0x57008000	.Trashes/501/.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;Removed;FileEvent;
302	0x14008000	.DS_Store	Modified;InodeMetaMod;FileEvent;

Trashed, Trash Emptied

Just Trashed

FSEvents Analysis: Volumes

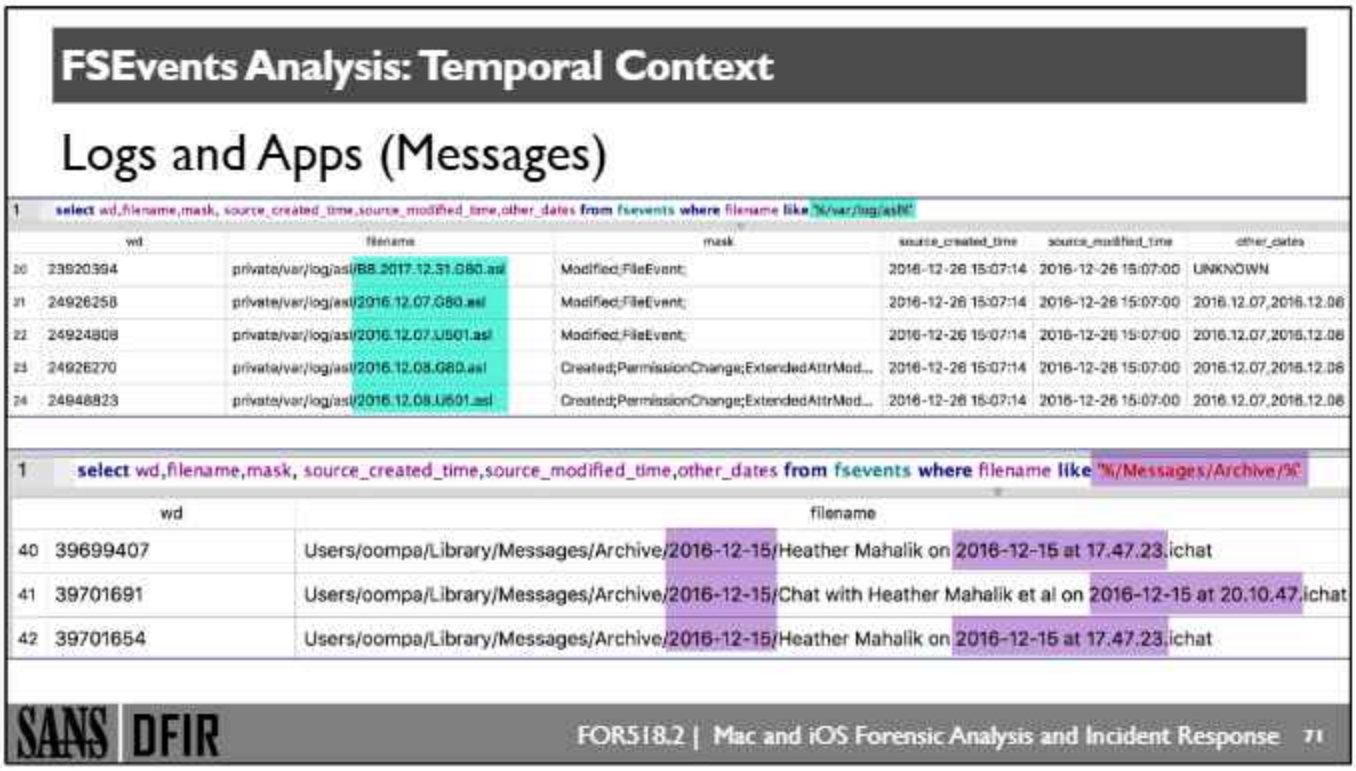
```
1  select wd, mask_hex,filename, mask, source from fsevents where filename like '%Volumes/Google%'
```

	wd	mask_hex	filename	mask	source
1	3829041	0x00000002	/Volumes/Google Chrome	Mount;	./00000000003af26b
2	3828991	0x80010001	Volumes/Google Chrome	FolderEvent;PermissionChange;FolderCreated;	./00000000003af26b
3	4511193	0x00000004	/Volumes/Google Chrome	Unmount;	./000000000045b9cb
4	4511185	0x02000001	Volumes/Google Chrome	FolderEvent;Removed;	./000000000045b9cb
5	29046995	0x00000002	/Volumes/Google Earth	Mount;	./0000000001bc10a7
6	29046984	0x80010001	Volumes/Google Earth	FolderEvent;PermissionChange;FolderCreated;	./0000000001bc10a7
7	30498055	0x00000004	/Volumes/Google Earth	Unmount;	./0000000001d2510c
8	30498053	0x02000001	Volumes/Google Earth	FolderEvent;Removed;	./0000000001d2510c

Other events can show system usage, such as "`Mount`" and "`Unmount`" volume activities.

The example above shows the file system events on a macOS image. Whenever a volume gets mounted onto "`/Volumes`", it is recorded. This example shows two volumes being mounted/unmounted: One for Google Chrome and another for Google Earth. It will show a mount point being created (`FolderCreated`), as well as removed after it is unmounted (`FolderEvent;Removed;`).

Reference:
https://github.com/dlcowen/FSEventsParser



All these file system events are being recorded; however, no timestamps are saved for each individual event. No problem—we'll just have to get creative!

The FSEvents Python script will show that the source created and modified timestamps: These are the file system timestamps of those fsevents gzip files. If the files are exported out of the image, they may show the wrong timestamps, as shown in the top screenshot; however, you can still go to the forensic image to get the correct timestamps. These timestamps can help filter down a scope of time the file activity occurred within.

The script also tries to guess timestamps by using timestamps within the filenames of nearby files, as shown in "`other_dates`". These timestamps are pulled from the ASL filenames shown in the top screenshot.

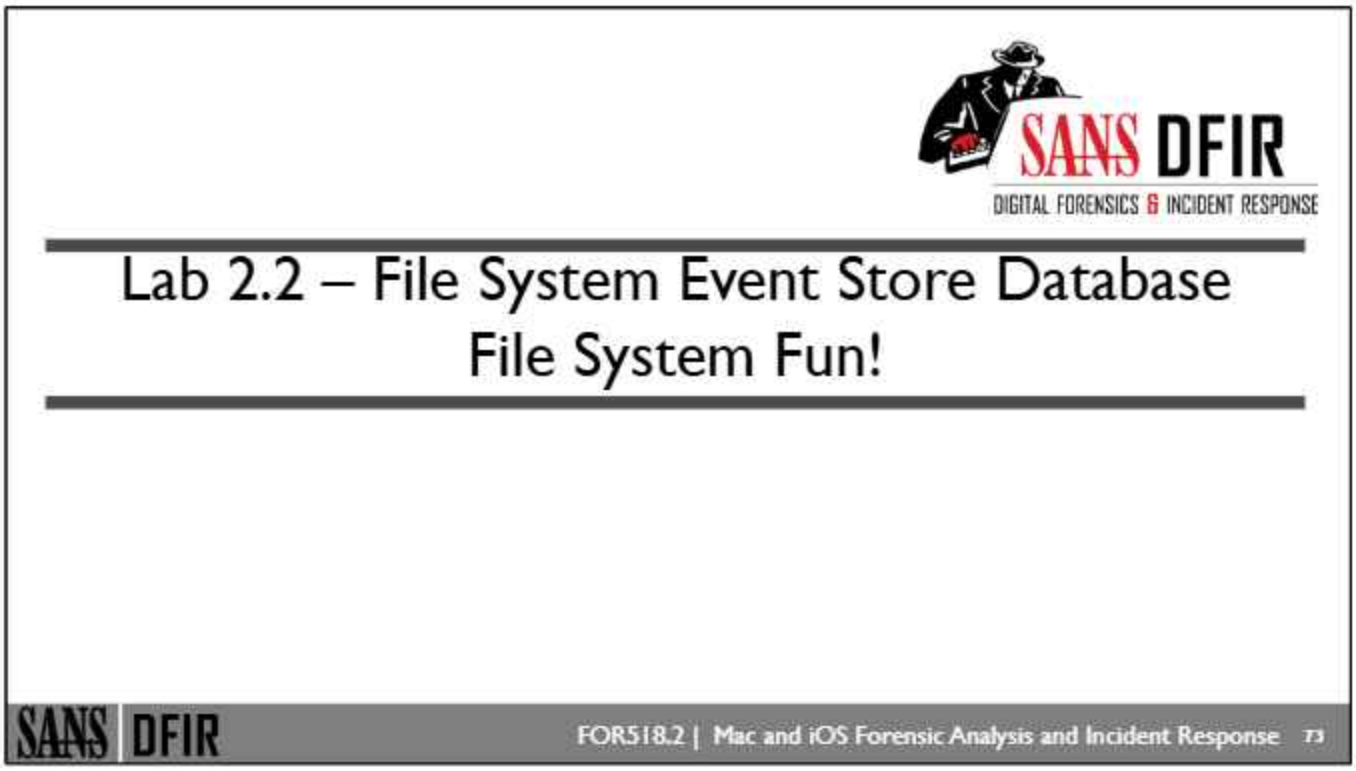
We can also use some timestamps file paths from other applications, assuming the user is using these applications.

The bottom screenshot shows chat messages being archived, which get recorded with a general timestamp of when the conversation occurred.

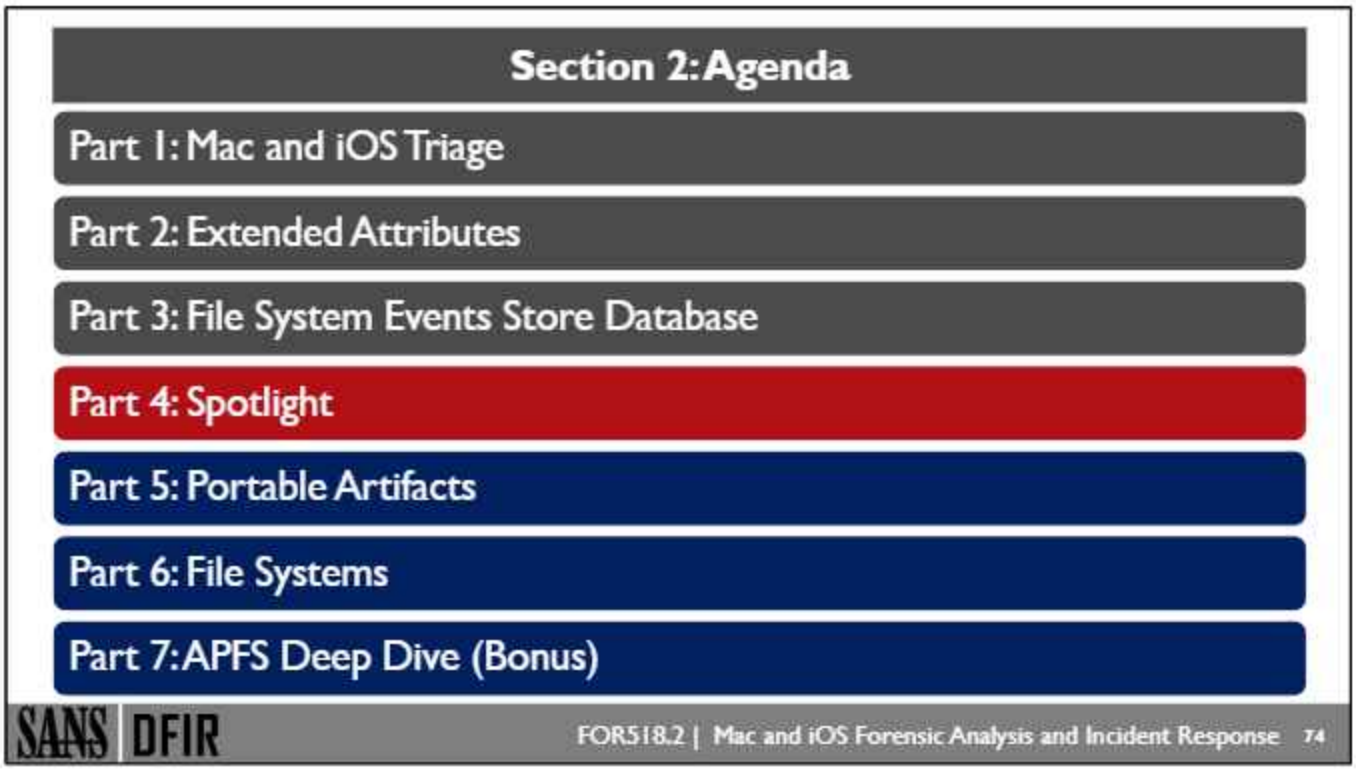
Reference:
https://github.com/dlcowen/FSEventsParser

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%/Photos Library.photoslibrary/Masters/2016/05/18%'
```

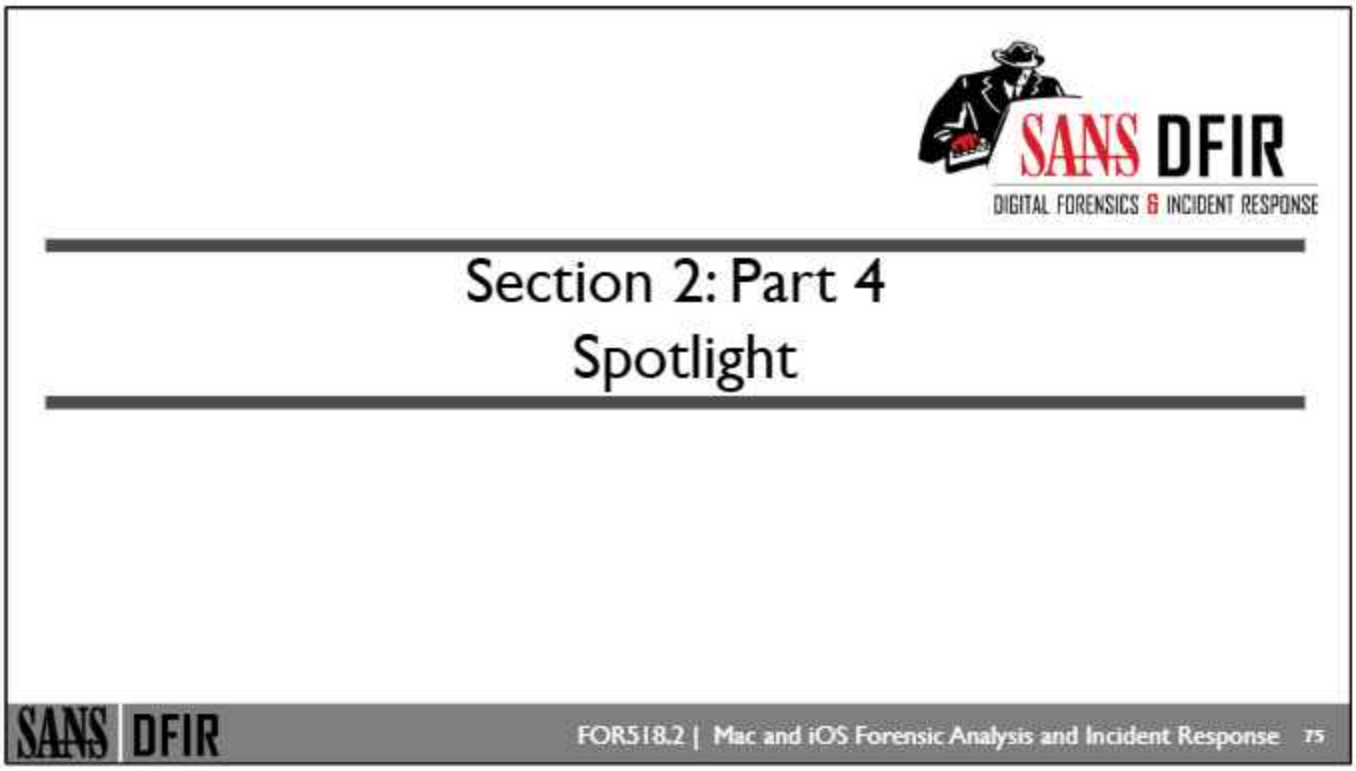
	wd	filename	mask
1	18158642773871313107	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18	FolderEvent;ExtendedAttrModified;FolderCre...
2	18158642773871314490	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-212119	FolderEvent;ExtendedAttrModified;FolderCre...
3	18158642773871314585	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-212119/IMG_0315.PNG	Modified;Created;HardLink;FileEvent;
4	18158642773871313113	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-220823	FolderEvent;ExtendedAttrModified;FolderCre...
5	18158642773871313788	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-220823/IMG_0316.PNG	Modified;Created;HardLink;FileEvent;
6	18158642773871313408	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757	FolderEvent;ExtendedAttrModified;FolderCre...
7	18158642773871313797	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757/IMG_0317.PNG	Modified;Created;HardLink;FileEvent;
8	18158642773871313800	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757/IMG_0318.PNG	Modified;Created;HardLink;FileEvent;



This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.

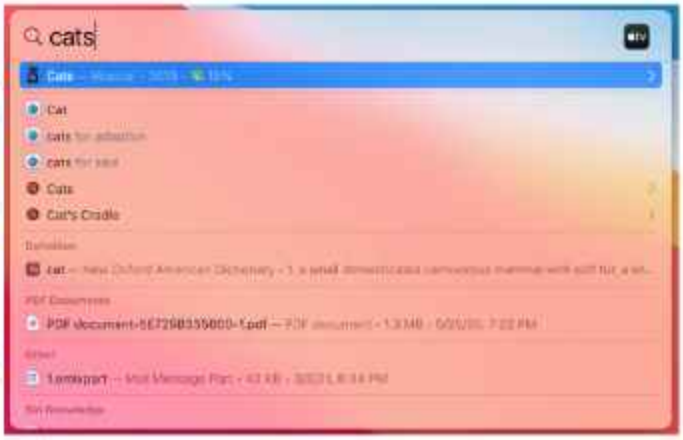
Spotlight

Desktop search system, introduced in 10.4

Indexed file metadata

Similar data found in extended attributes

Writable internal/external media indexed on per-volume basis

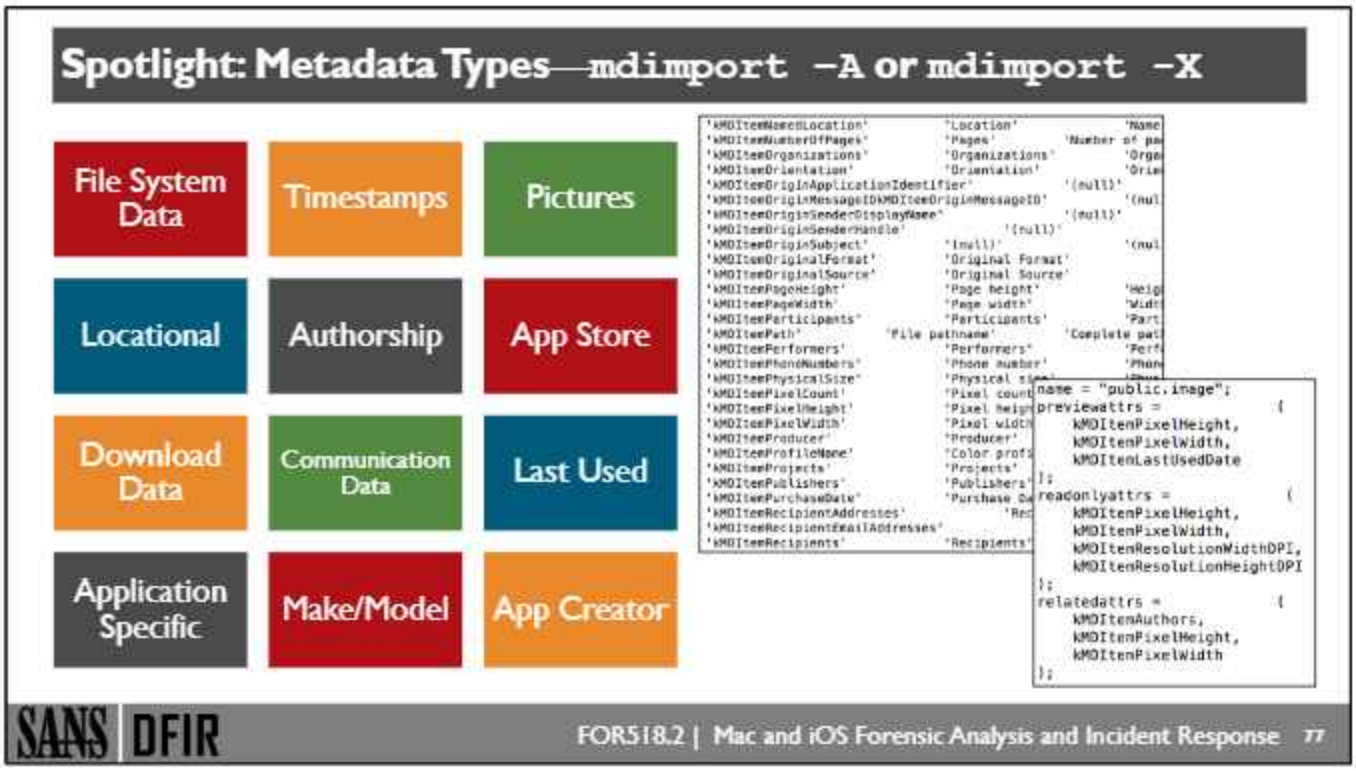


SANS DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 76

macOS uses Spotlight to index the system to allow the user to search for applications, documents, and other files quickly. This feature was introduced in OS X 10.4 and on iPhones with iOS 3. File metadata, including the extended attributes, are all indexed. On a live system, the Spotlight icon in the top right of the screen can be used to search the Spotlight index.

While system hard drives and external drives are indexed, other locations may not be indexed by default. These locations include DMG files, CDs and DVDs, hidden files, and system directories. A volume can explicitly be told not to be indexed by placing an empty, hidden file with the filename `.metadata_never_index` at the root of the volume. A volume can be set not to index by using the `mdutil` command.

References:
Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin—Chapter 2: The User Experience Layer
`mdutil` Man Page



What type of data gets indexed? Anything and everything!

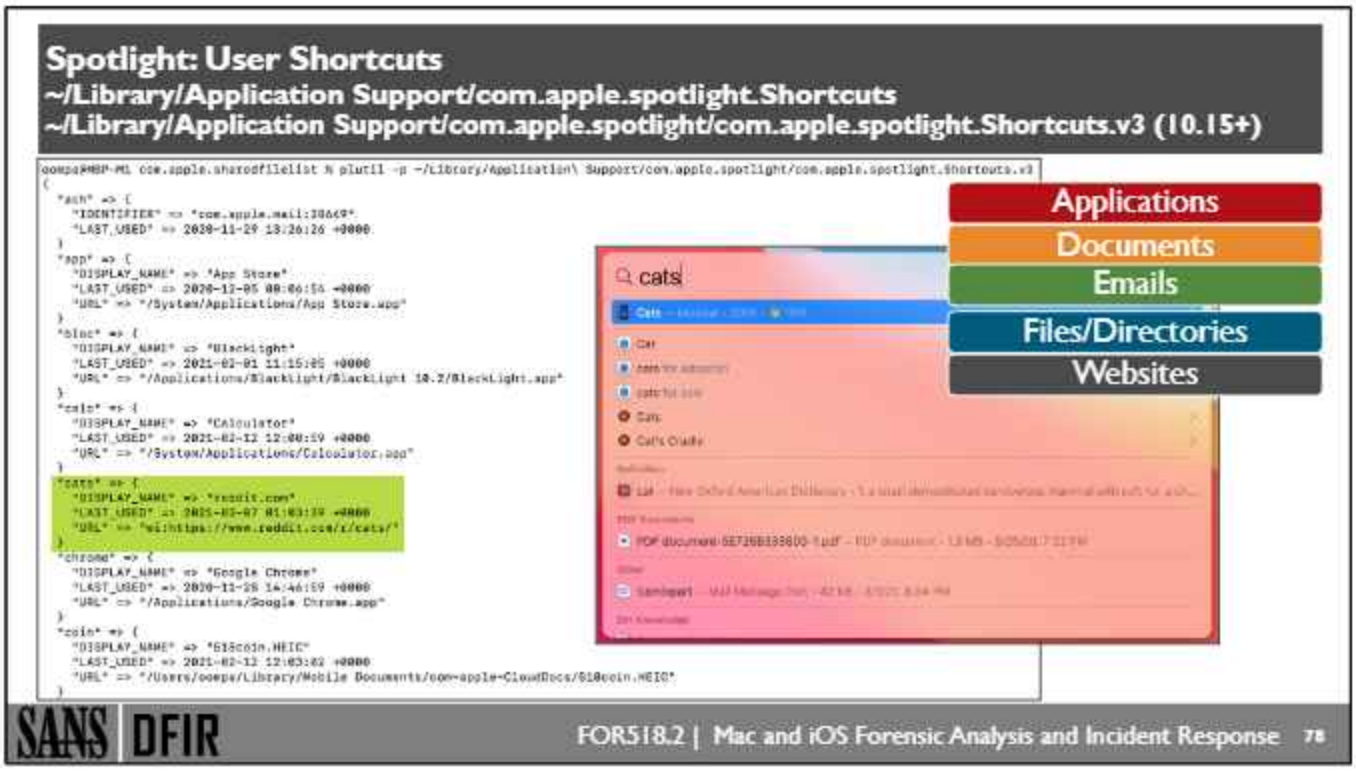
A small example of what might be indexed:

- Another copy of file system metadata, including filenames, logical and physical file sizes, UID/GID, and file system timestamps.
- Timestamps: Along with the file system timestamps, you may also find download dates, last used dates, and date added dates.
- Photos data may include width/length, make and model of hardware that took the photo, aperture, ISO speeds, pixel count, and resolution.
- Authorship information may include who originated the file and what type of application created the document.
- If an application was downloaded or purchased through the Apple App Store, certain receipt data will be indexed.
- Communication information such as what email address sent an attachment or what the hostname was of the phone that AirDropped a photo to the system.
- Depending on what software is installed on the system, certain applications may have their own metadata attributes.

Two native command line utilities can show us what type of data may be indexed. These commands will only show the attributes that are associated with the host system, not the mounted system. Many of the `kMD*` attributes will be the same across systems; but keep an eye out for those application-specific metadata attributes. Newer systems will generally contain more attributes than older systems.

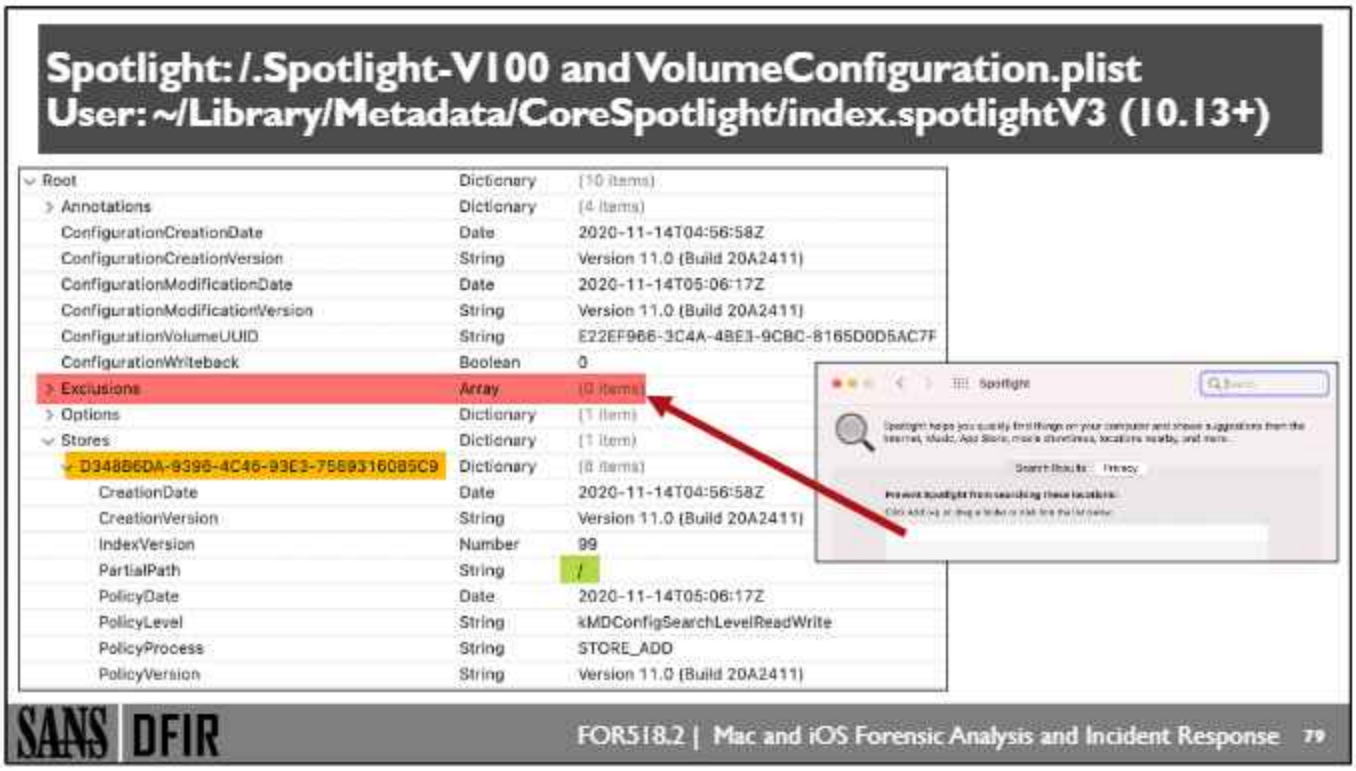
The "`mdimport -A`" command (shown on the left) will print out the attributes' names and descriptions. The "`mdimport -X`" command (shown on the right) prints out metadata attributes associated with a specific file type; for example, attributes associated with photos.

Reference:
Man Page for `mdimport`



Spotlight is an application that is used to index and create a searchable database of the macOS system. Users can search for items using the Spotlight magnifying glass icon in the top right of the GUI or using Cmd+Spacebar.

These are the first few letters the user typed before selecting an item in the dropdown. If the user types the same few letters again and selects something, the timestamp (and possibly the path) will be updated; otherwise, it is a historical listing of Spotlight searches.



The hidden `.Spotlight-V100` directory located in the root of the volume contains the Spotlight Store directories.

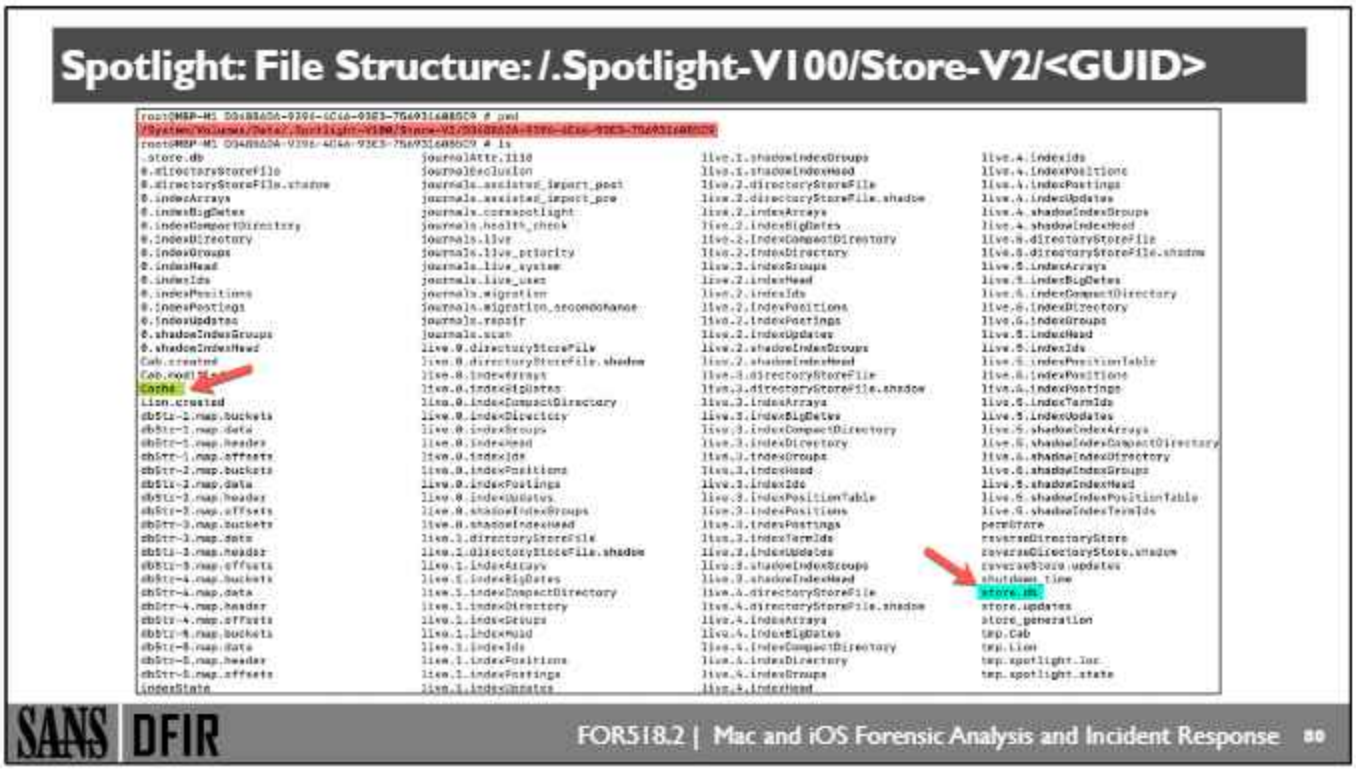
The `.Spotlight-V100` directory contains the property list `VolumeConfiguration.plist`. This property list contains indexing exclusions (among other Spotlight configuration data).

The single GUID in the example shows that one volume is currently being indexed:

- / - The root volume

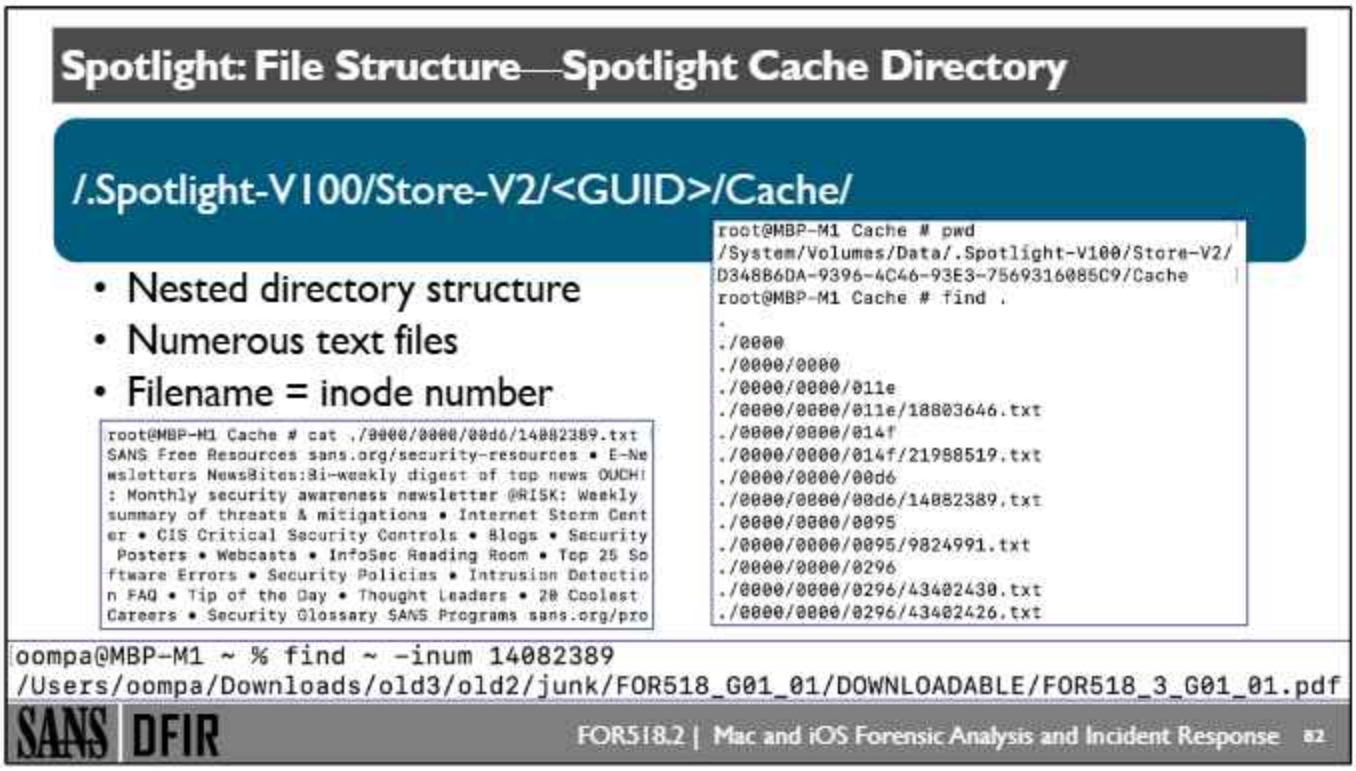
The GUID references the Spotlight stores where the volume indexing database (store.db) is stored.

Starting in 10.13, each user has their own Spotlight database located in their ~/Library directory.
http://www.swiftforensics.com/2018/10/the-user-spotlight-database.html



Each Spotlight Store directory contains many files. These files are the indexing databases used by Spotlight. The two places where data is readily available for investigators are the `/Cache` directory and the `store.db` index database.

```
root@MBP-M1 D348B6DA-9396-4C46-93E3-756931608SC9 # pwd
/System/Volumes/Data/.Spotlight-V100/Store-V2/D348B6DA-9396-4C46-93E3-756931608SC9
root@MBP-M1 D348B6DA-9396-4C46-93E3-756931608SC9 # ls
.store.db                          journalAttr.1118                   live.1.shadowIndexGroups           live.4.indexIds
0.directoryStoreFile               journalExclusion                   live.1.shadowIndexHead             live.4.indexPositions
0.directoryStoreFile.shadow        journals.assisted_import_post      live.2.directoryStoreFile          live.4.indexPostings
0.indexArrays                      journals.assisted_import_pre       live.2.directoryStoreFile.shadow   live.4.indexUpdates
0.indexBigDates                    journals.corespotlight             live.2.indexArrays                 live.4.shadowIndexGroups
0.indexCompactDirectory            journals.health_check              live.2.indexBigDates               live.4.shadowIndexHead
0.indexDirectory                   journals.live                      live.2.indexCompactDirectory       live.5.directoryStoreFile
0.indexGroups                      journals.live_priority             live.2.indexDirectory              live.5.directoryStoreFile.shadow
0.indexHead                        journals.live_system               live.2.indexGroups                 live.5.indexArrays
0.indexIds                         journals.live_user                 live.2.indexHead                   live.5.indexBigDates
0.indexPositions                   journals.migration                 live.2.indexIds                    live.5.indexCompactDirectory
0.indexPostings                    journals.migration_secondchance    live.2.indexPositions              live.5.indexDirectory
0.indexUpdates                     journals.repair                    live.2.indexPostings               live.5.indexGroups
0.shadowIndexGroups                journals.scan                      live.2.indexUpdates                live.5.indexHead
0.shadowIndexHead                  live.0.directoryStoreFile          live.2.shadowIndexGroups           live.5.indexIds
Cab.created                        live.0.directoryStoreFile.shadow   live.2.shadowIndexHead             live.5.indexPositionTable
Cab.modified                       live.0.indexArrays                 live.3.directoryStoreFile          live.5.indexPositions
Cache                              live.0.indexBigDates               live.3.directoryStoreFile.shadow   live.5.indexPostings
Lion.created                       live.0.indexCompactDirectory       live.3.indexArrays                 live.5.indexTermIds
dbStr-1.map.buckets                live.0.indexDirectory              live.3.indexBigDates               live.5.indexUpdates
dbStr-1.map.data                   live.0.indexGroups                 live.3.indexCompactDirectory       live.5.shadowIndexArrays
dbStr-1.map.header                 live.0.indexHead                   live.3.indexDirectory              live.5.shadowIndexCompactDirectory
dbStr-1.map.offsets                live.0.indexIds                    live.3.indexGroups                 live.5.shadowIndexDirectory
dbStr-2.map.buckets                live.0.indexPositions              live.3.indexHead                   live.5.shadowIndexGroups
dbStr-2.map.data                   live.0.indexPostings               live.3.indexIds                    live.5.shadowIndexHead
dbStr-2.map.header                 live.0.indexUpdates                live.3.indexPositionTable          live.5.shadowIndexPositionTable
dbStr-2.map.offsets                live.0.shadowIndexGroups           live.3.indexPositions              live.5.shadowIndexTermIds
dbStr-3.map.buckets                live.0.shadowIndexHead             live.3.indexPostings               permStore
dbStr-3.map.data                   live.1.directoryStoreFile          live.3.indexTermIds                reverseDirectoryStore
dbStr-3.map.header                 live.1.directoryStoreFile.shadow   live.3.indexUpdates                reverseDirectoryStore.shadow
dbStr-3.map.offsets                live.1.indexArrays                 live.3.shadowIndexGroups           reverseStore.updates
dbStr-4.map.buckets                live.1.indexBigDates               live.3.shadowIndexHead             shutdown_time
dbStr-4.map.data                   live.1.indexCompactDirectory       live.4.directoryStoreFile          store.db
dbStr-4.map.header                 live.1.indexDirectory              live.4.directoryStoreFile.shadow   store.updates
dbStr-4.map.offsets                live.1.indexGroups                 live.4.indexArrays                 store_generation
dbStr-5.map.buckets                live.1.indexHead                   live.4.indexBigDates               tmp.Cab
dbStr-5.map.data                   live.1.indexIds                    live.4.indexCompactDirectory       tmp.Lion
dbStr-5.map.header                 live.1.indexPositions              live.4.indexDirectory              tmp.spotlight.loc
dbStr-5.map.offsets                live.1.indexPostings               live.4.indexGroups                 tmp.spotlight.state
indexState                         live.1.indexUpdates                live.4.indexHead
```



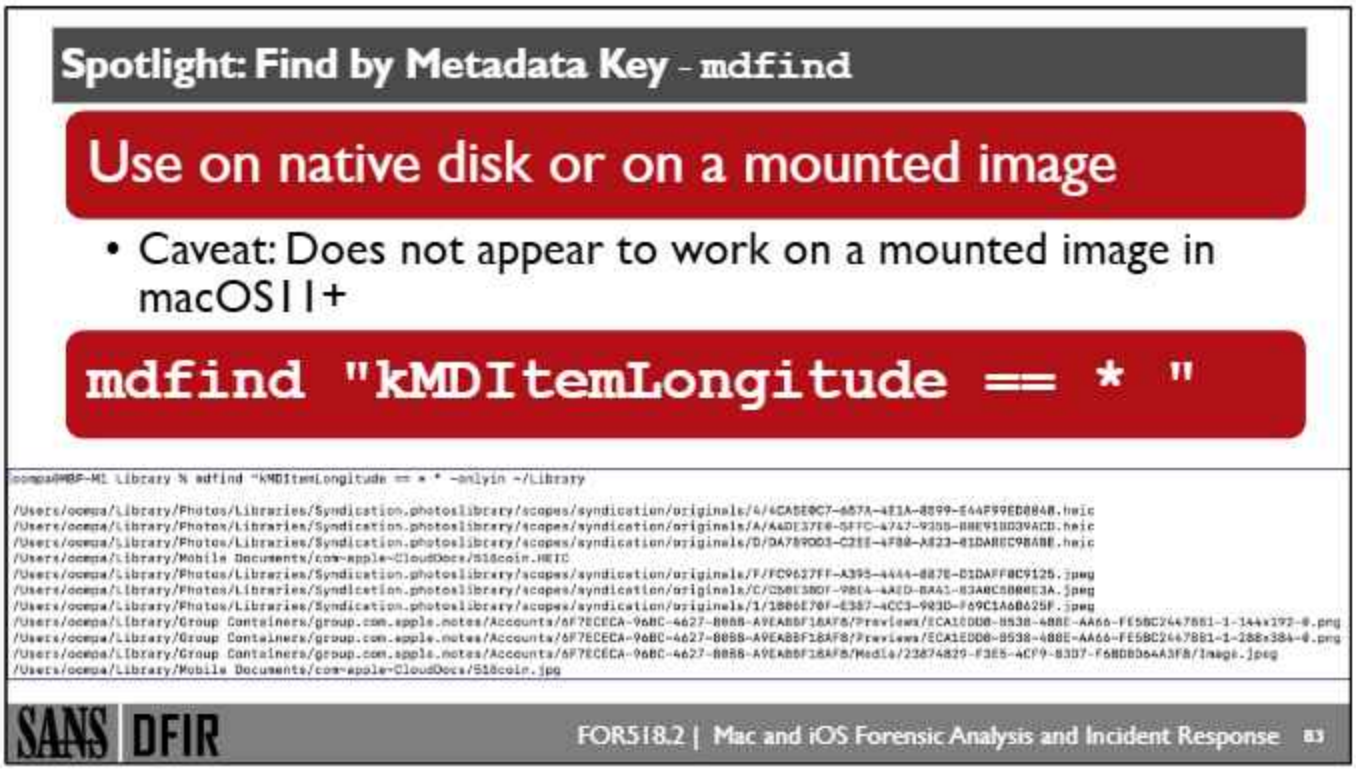
One directory that can be of use to forensic analysts is the Spotlight Cache directory. This directory contains a nesting of other subdirectories containing many text files. These text files hold text-based versions of their original documents, including emails, documents, and chats.

The number preceding each `.txt` extension is the inode number for the file it represents. The lower screenshot shows an example of this. The filename 14082389`.txt` contains the email-related data.

We can find the associated file by searching the system for the inode, 14082389. We can use the `find` command on the mounted volume. Use the `-inum` argument to find the file by inode number.

```
sudo find <volume> -inum 14082389
```

Experimentation shows these text files will no longer be available when a file is deleted or simply moved to the trash.



The "mdfind" command will find files based on certain metadata criteria.

In the example, the attribute "kMDItemLongitude" was searched for. This will print out the filenames and paths for all files on the system (or mounted image) where this attribute was indexed. This example will show us files that contain locational data.

The -onlyin argument does not currently work on macOS 11 or newer.

Reference:
Man Page for mdfind

Spotlight: List Metadata - `mdls`

Use on native disk or on a mounted image

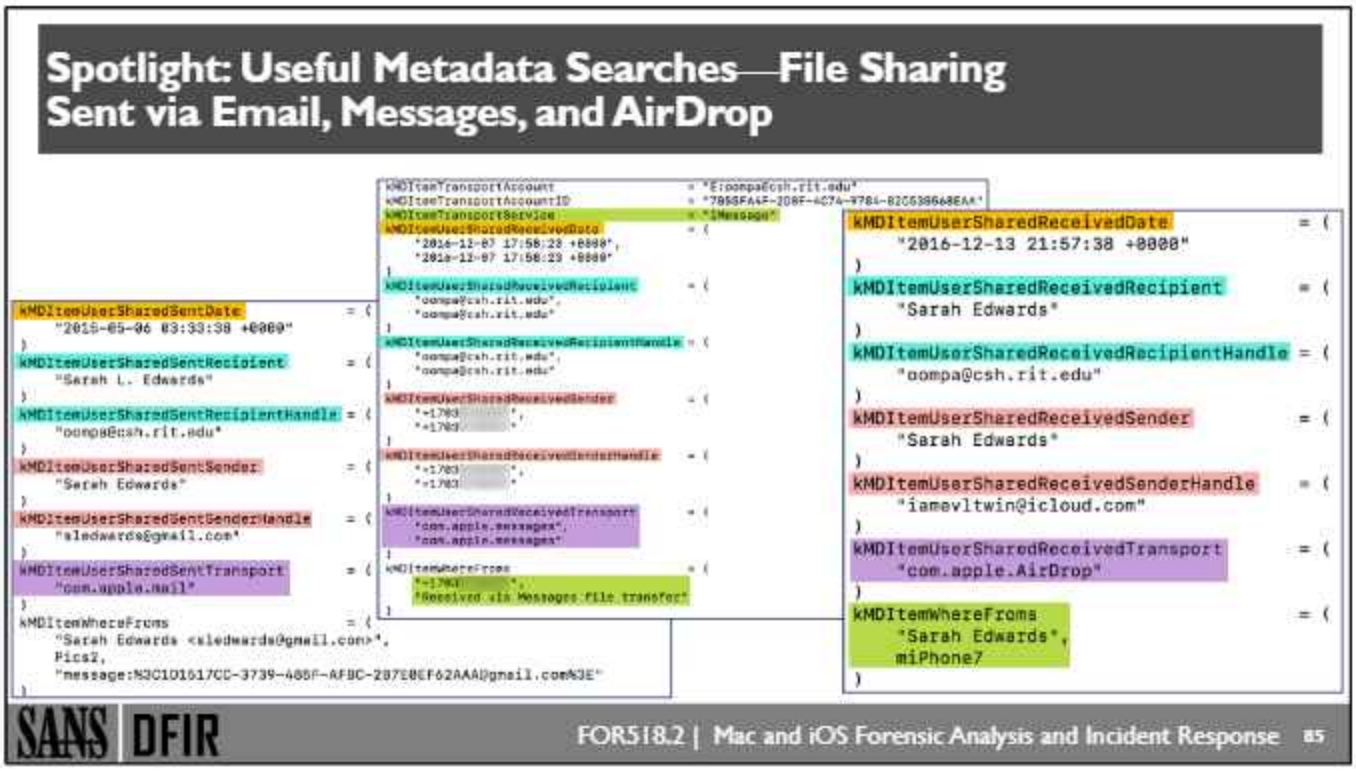
- Caveat: Does not appear to work on a mounted image in macOS11+

The `mdls` command can be used to list the metadata associated with a particular file from the Spotlight databases. In the screenshot above, we can see the following attributes of the file:

- Timestamps
- Content Types
- Download Date
- Download Location
- File Sizes
- File Ownership
- File Properties

This screenshot is only a partial output of the metadata associated with this file. The complete output of this command is shown on the next page.

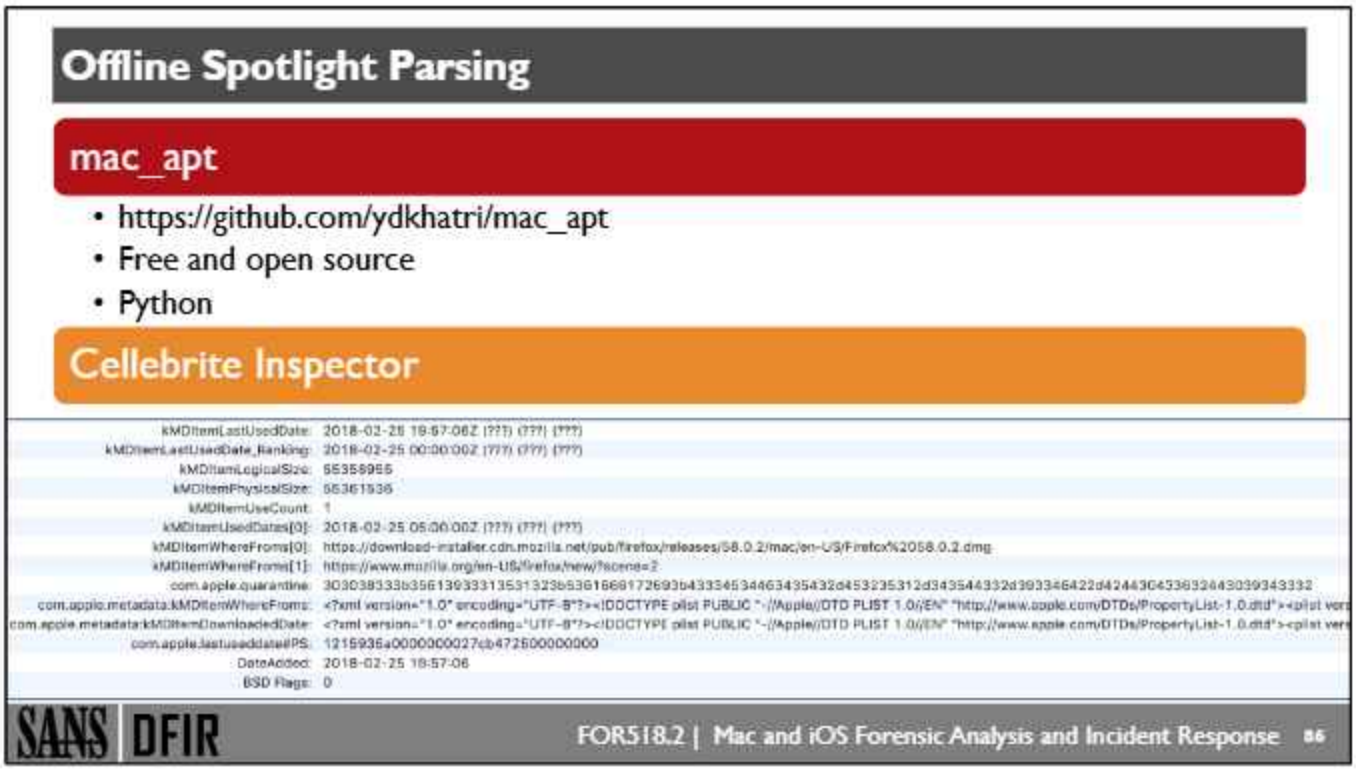
Reference:
Man Page for `mdls`



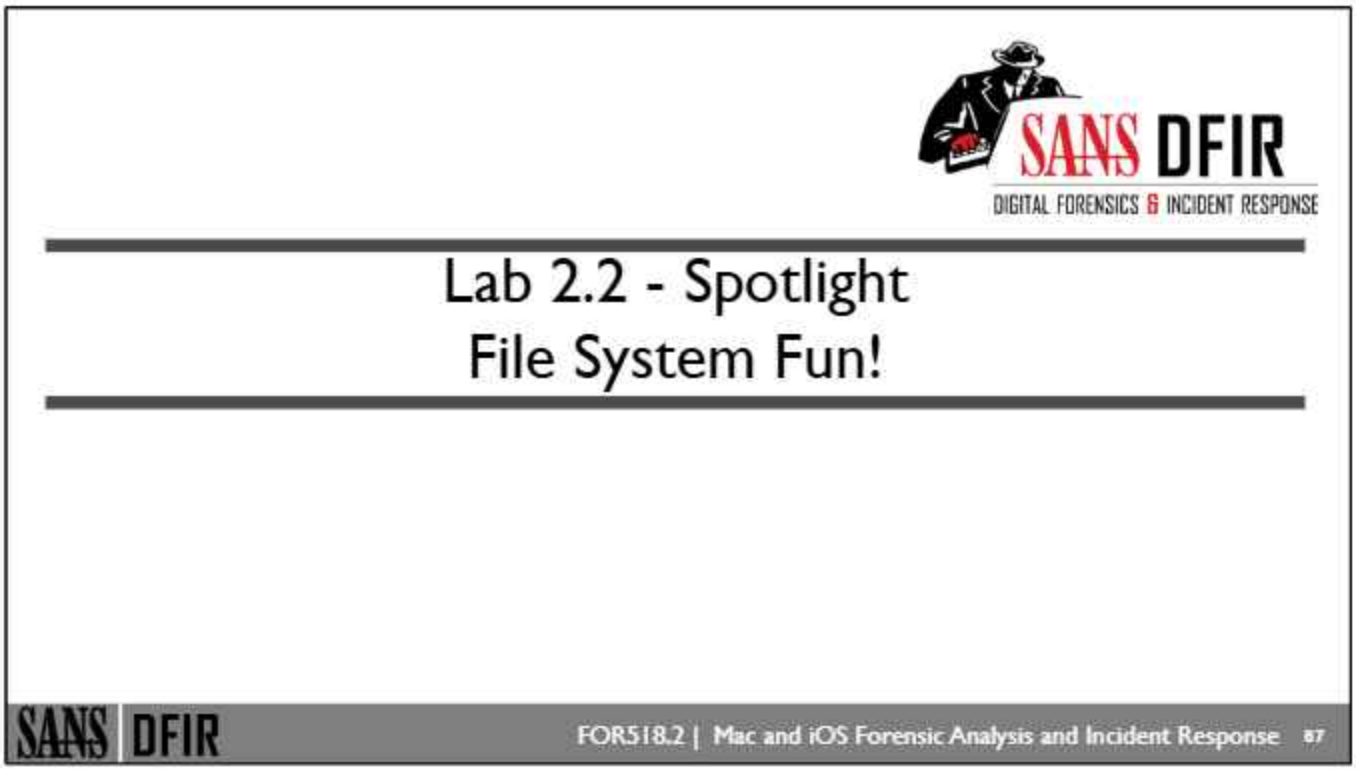
On newer operating systems, there are additional sharing metadata attributes that can be forensically very interesting and useful. Using `mdfind` to query files that contain the `kMDItemUserSharedSentDate` attribute, we can find files that have been shared out. The screenshot on the left shows an example of a file sent via Apple Mail (com.apple.mail) on 05/06/2015. This email had the subject "Pics2" and was sent from sledwards@gmail.com to oompa@csh.rit.edu.

On the flip side, we can also use `mdfind` to query for files that have been received by the system using "`kMDItemUserSharedReceivedDate`". The example in the middle shows a file received via the Messages application (com.apple.messages) from a phone number "+1703 ..." to oompa@csh.rit.edu on 12/07/2016. This file was received using the iMessage protocol in the Messages application.

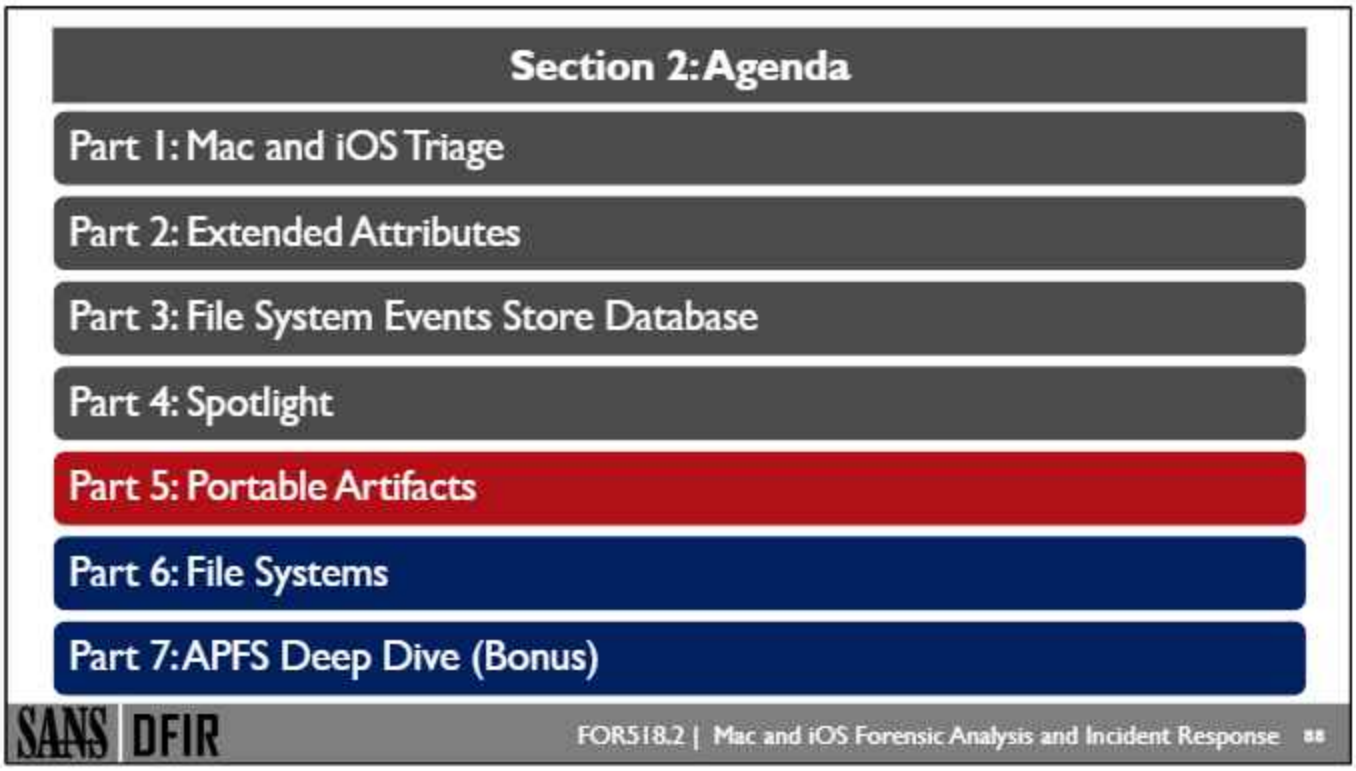
Files received via the AirDrop service can be hard to discern without digging a bit further. These newer sharing metadata attributes can make this easier to determine. The screenshot on the right shows a file was received on 12/13/2016 from iamevltwin@icloud.com to oompa@csh.rit.edu using AirDrop (com.apple.AirDrop). The `kMDItemWhereFroms` attribute gives us the username and device name that were used to AirDrop from. In this case, it was Sarah's "miPhone7". The bottom right screenshot shows that it was AirDropped from the com.apple.mobileslideshow application on Sarah's miPhone7. This is the Photos application on the iPhone.



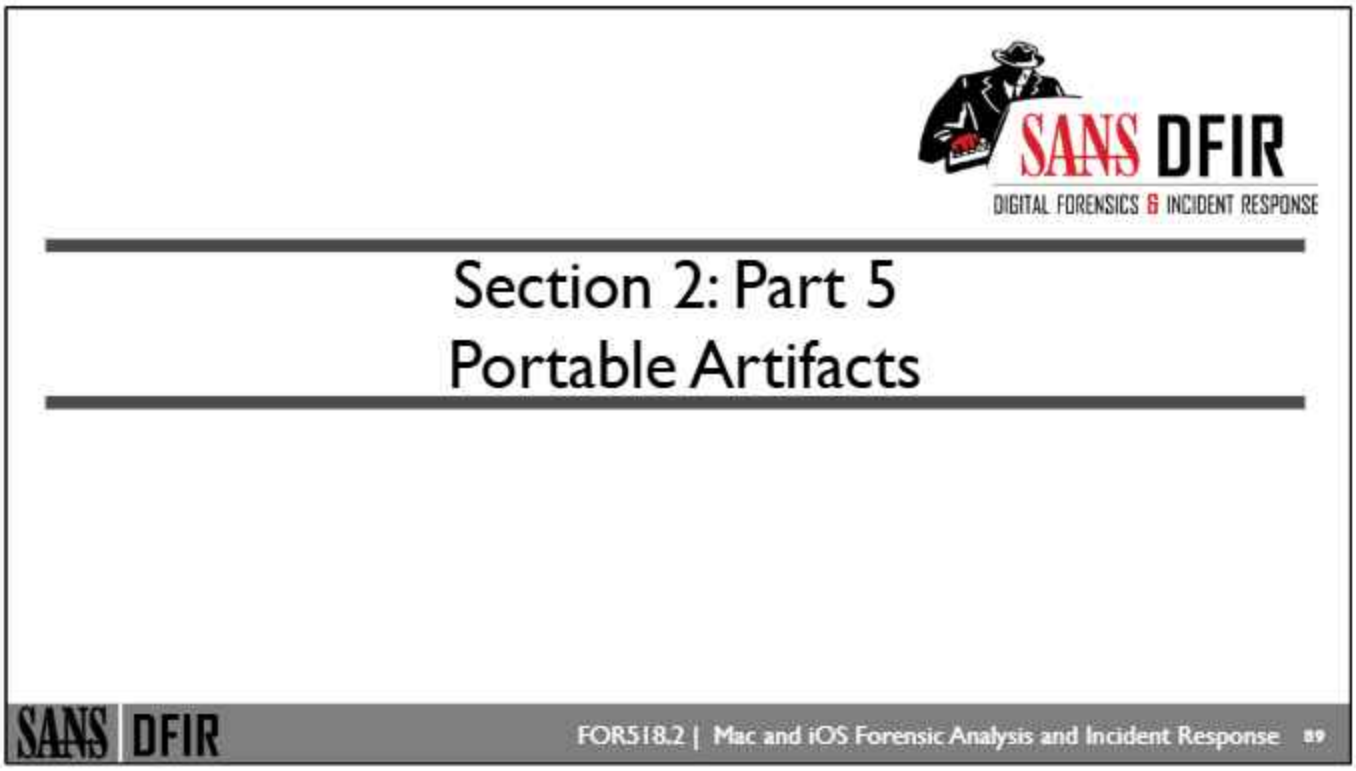
There are a few tools that have the capability to parse the store.db files offline (versus mounting and live queries). The first is the open-source mac_apt utility that is a multiplatform Python script, while the second is Inspector (using the Advanced Parsing).



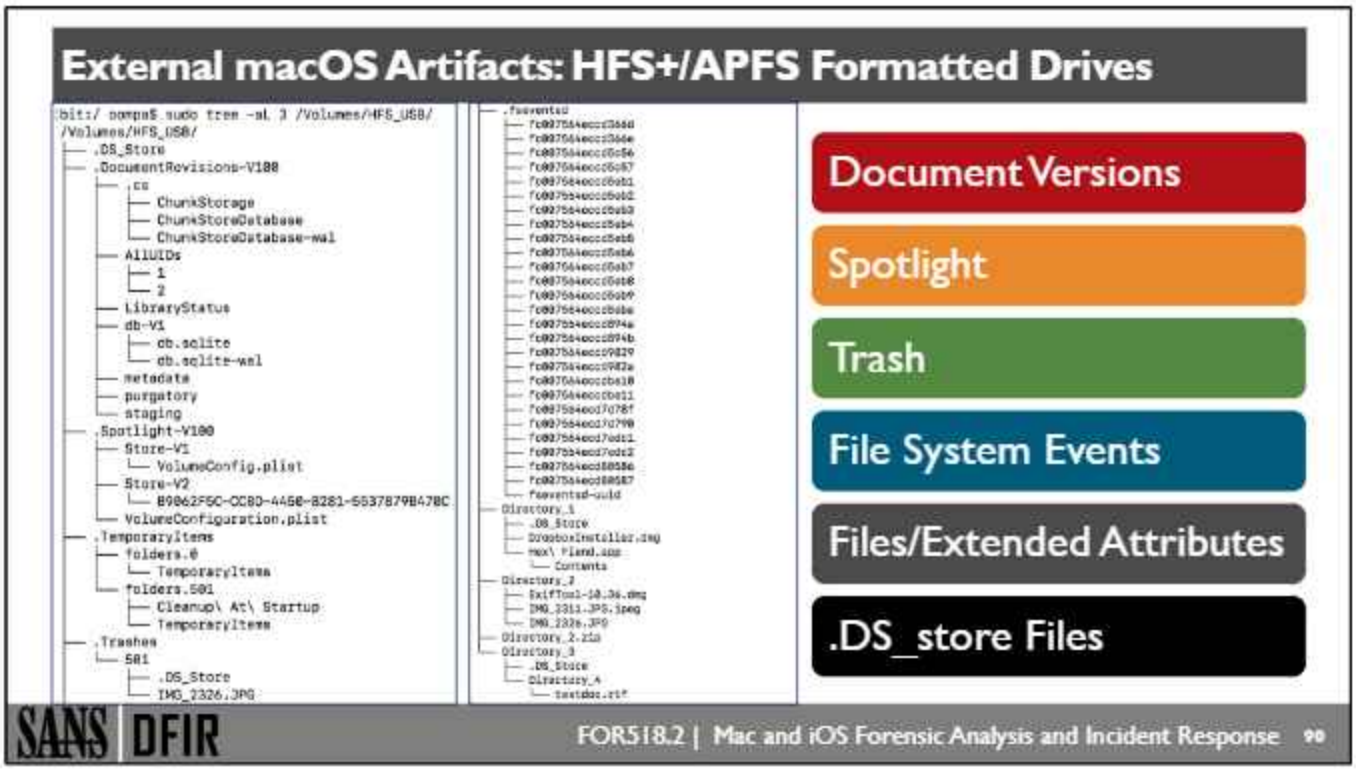
This page intentionally left blank.



This page intentionally left blank.



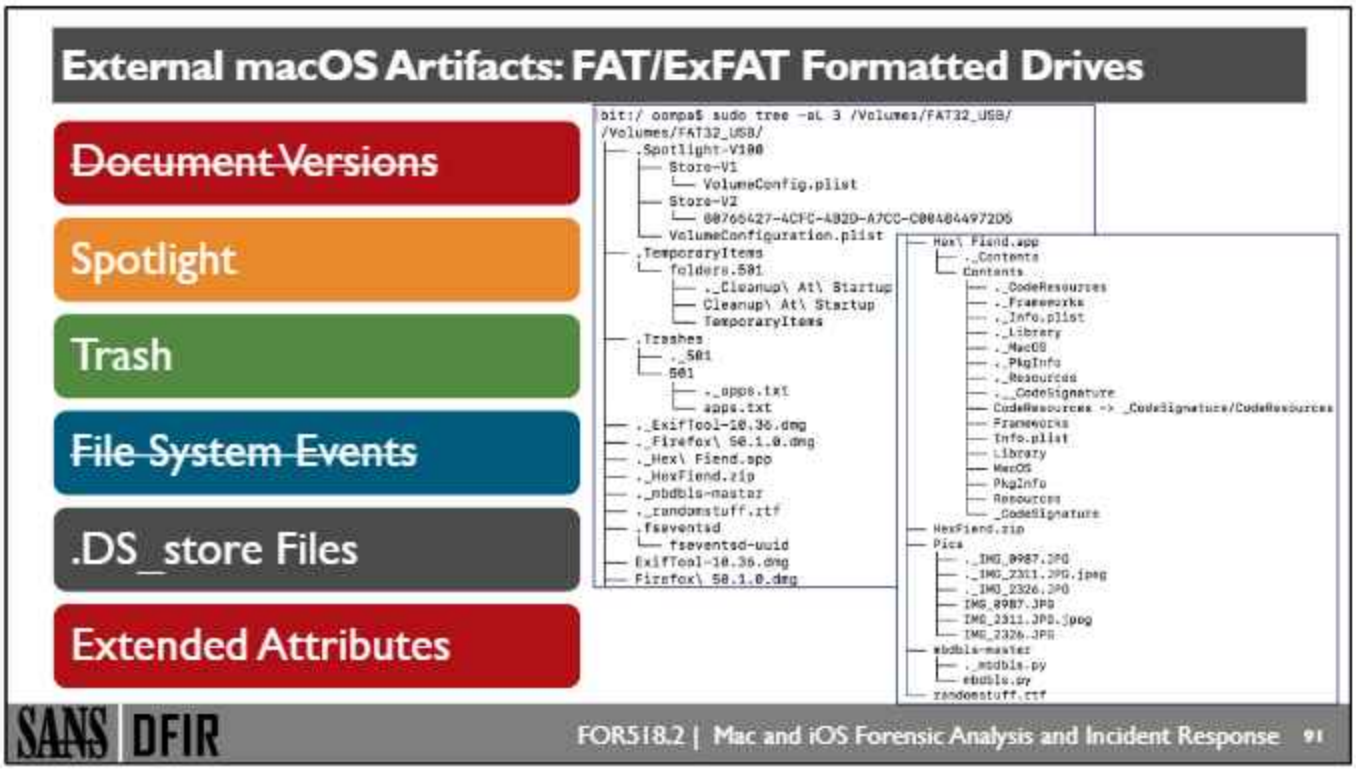
This page intentionally left blank.



macOS can leave behind some interesting artifacts that may be shown on other non-Mac systems. These artifacts can help us determine if a file was originally found on a Mac system. These artifacts may include various directories, extended attributes, and those sneaky `.DS_store` files that seem to get everywhere!

Files copied on an HFS+/APFS formatted drive will show signs of Mac origination, including the following:

- Extended Attributes
- Spotlight Index
- Document Versions
- Trashes Directory
- `.fseventsd` Directory
- `.DS_store` Files



Files copied to FAT or ExFAT formatted drives will show signs of macOS originations, including a Spotlight directory (with `store.db` Spotlight index!) and a `.Trashes` directory if files were "removed".

Extended attribute information is also relocated along with each file; however, FAT-based file systems have no support for them, and therefore the data needs to be stored differently. Each file that has extended attributes will have an additional file beginning with a "`._`" that matches the name of the file. This contains the extended attribute information, as shown in the next slide.

Unfortunately, FAT and ExFAT drives will not have the Document Revisions directories, as these volumes do not support Document Versions capability. Along the same line, the `.fseventsd` directory will exist, but no files appear to be written into it.

FAT Formatted Drives: Extended Attributes

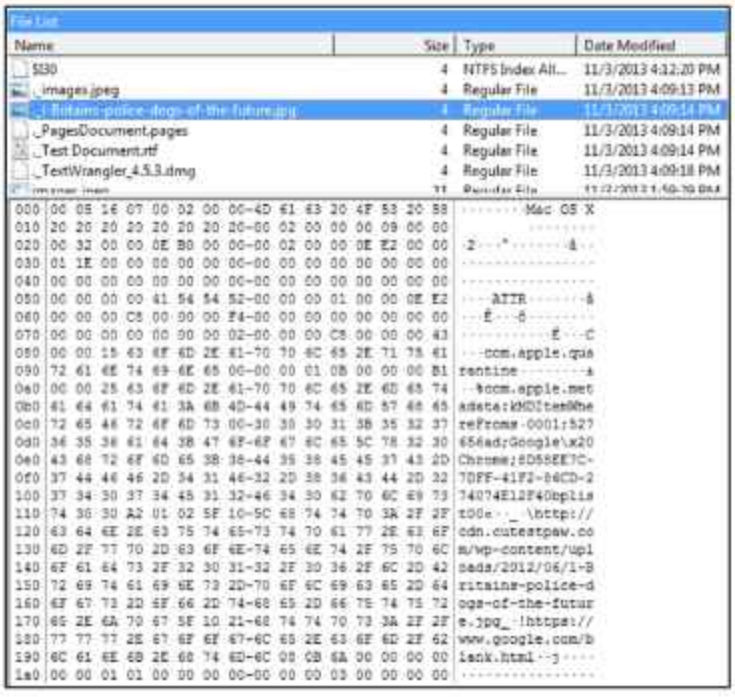
```
bit:FAT32_USB oompa$ xxd ._HexFiend.zip
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0e2e 0000 0032 0000 0e2e 0000  .2.......2......
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000100: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000110: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000120: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000130: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000140: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000150: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000160: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000170: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000180: 0000 0000 0000 0300 0000 0000 0000 0000  ................
00000190: 0000 0000 0000 6900 0000 0000 0000 0000  ......i.........
000001a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

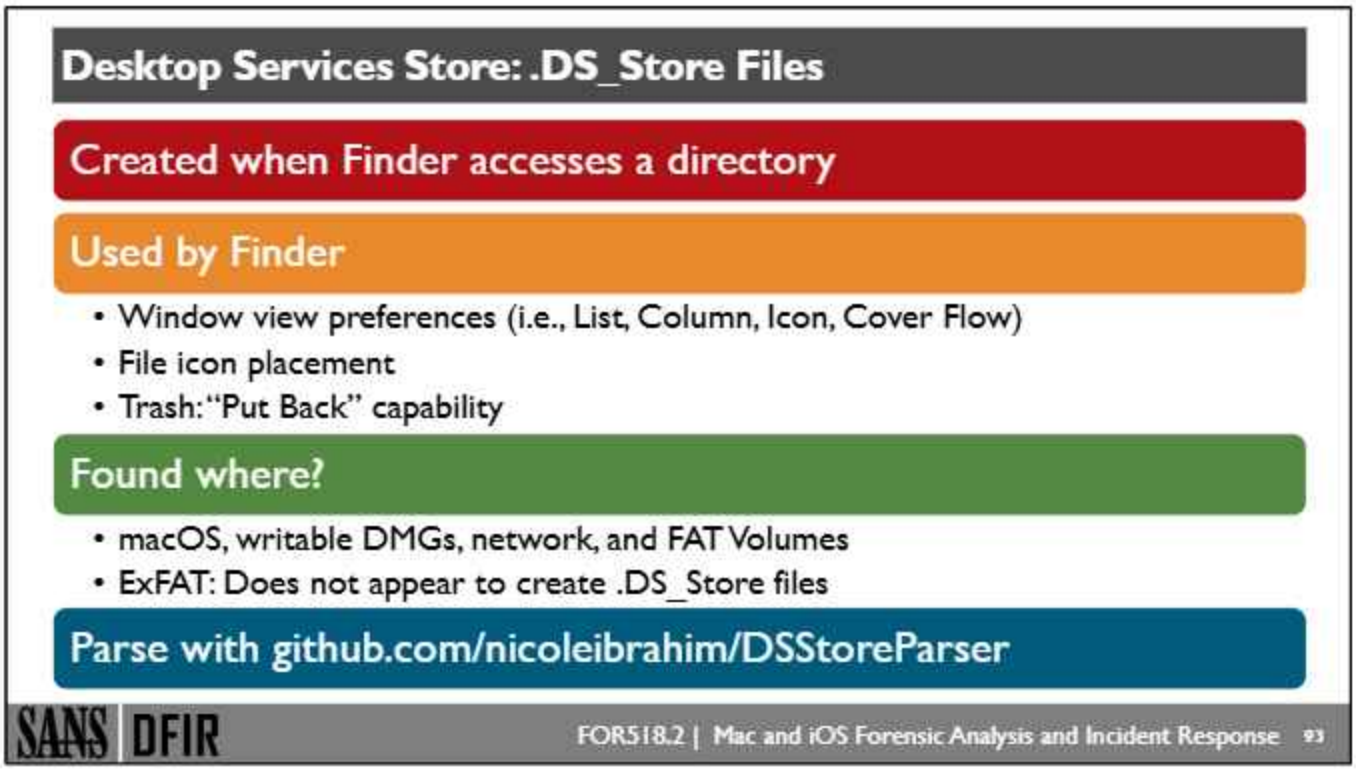
```
bit:FAT32_USB oompa$ xattr -xl  HexFiend.zip
com.apple.quarantine:
00000000  30 30 38 31 3B 35 38 34 38 63 37 65 35 3B 47 6F  |0081;5848c7e5;Go|
00000010  6F 67 6C 65 5C 78 32 30 43 68 72 6F 6D 65 3B 34  |ogle\x20Chrome;4|
00000020  30 37 35 31 33 37 43 2D 31 39 33 44 2D 34 44 39  |075137C-193D-4D9|
00000030  42 2D 42 41 32 46 2D 46 36 34 42 44 38 46 39 46  |B-BA2F-F64BD8F9F|
00000040  31 32 32                                         |122|
00000043
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 35 68 74  |bplist00..._.5ht|
00000010  74 70 3A 2F 2F 72 69 64 69 63 75 6C 6F 75 73 66  |tp://ridiculousf|
00000020  69 73 68 2E 63 6F 6D 2F 68 65 78 66 69 65 6E 64  |ish.com/hexfiend|
00000030  2F 66 69 6C 65 73 2F 48 65 78 46 69 65 6E 64 2E  |/files/HexFiend.|
00000040  7A 69 70 5F 10 23 68 74 74 70 3A 2F 2F 72 69 64  |zip_.#http://rid|
00000050  69 63 75 6C 6F 75 73 66 69 73 68 2E 63 6F 6D 2F  |iculousfish.com/|
00000060  68 65 78 66 69 65 6E 64 2F 08 0B 43 00 00 00 00  |hexfiend/..C....|
00000070  00 00 01 01 00 00 00 00 00 00 00 03 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00 00 00 00 69              |...........i|
0000008c
```

```
bit:FAT32_USB oompa$ xxd ._HexFiend.zip
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0e2e 0000 0032 0000 0e2e 0000  .2.......2......
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ....ATTR;.......
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0000 0000 0000 0000 0000 0000 0000 0043  ...............C
00000070: 0000 0000 0000 0000 0000 0000 0000 0000  ...com.apple.qua
00000080: 0000 0000 0000 0000 0000 0000 0000 0000  rantine.........
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  ..%com.apple.met
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000  adata:kMDItemWhe
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000  reFroms.0081;584
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000  8c7e5;Google\x20
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000  Chrome;4075137C-
000000e0: 0000 0000 0000 0000 0000 0000 0000 0000  193D-4D99-BA2F-F
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  64BD8F9F122bplis
00000100: 0000 0000 0000 0300 0000 0000 0000 0000  t00..._.5http://
00000110: 0000 0000 0000 0000 0000 0000 0000 0000  ridiculousfish.c
00000120: 0000 0000 0000 0000 0000 0000 0000 0000  om/hexfiend/file
00000130: 0000 0000 0000 0000 0000 0000 0000 0000  s/HexFiend.zip_.
00000140: 0000 0000 0000 0000 0000 0000 0000 0000  #http://ridiculo
00000150: 0000 0000 0000 0000 0000 0000 0000 0000  usfish.com/hexfi
00000160: 0000 0000 0000 0000 0000 0000 0000 0000  end/..C.........
00000170: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000180: 0000 0000 0000 0300 0000 0000 0000 0000  ................
00000190: 0000 0000 0000 6900 0000 0000 0000 0000  ......i.........
000001a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```


Extended attributes that are normally stored in the Attributes File on HFS+ file systems are stored in separate files on FAT32 and ExFAT file systems. These files share the same filename as the associated file with a "dot underscore" (`._`) appended to the beginning. These files are 4,096 bytes in size.

Shown in the larger screenshot are the file signatures "`Mac OS X`" and "`ATTR`", followed by the contents of the extended attribute. At the end of the file is the text "`This resource fork intentionally left blank`" (not shown in screenshot). The format of these "`._`" files contains the attribute names first, followed by the data. The inset screenshot shows what the data will look like if printed with the "`xattr`" command.


File List
Name
Size
Type
Date Modified
$I30
4
NTFS Index All...
11/3/2013 4:12:20 PM
._images.jpeg
4
Regular File
11/3/2013 4:09:13 PM
._Britains-police-dogs-of-the-future.jpg
4
Regular File
11/3/2013 4:09:14 PM
._PagesDocument.pages
4
Regular File
11/3/2013 4:09:14 PM
._Test Document.rtf
4
Regular File
11/3/2013 4:09:14 PM
._TextWrangler_4.5.3.dmg
4
Regular File
11/3/2013 4:09:18 PM
000 00 05 16 07 00 02 00 00-4D 61 63 20 4F 53 20 58 ····· Mac OS X
010 20 20 20 20 20 20 20 20-00 02 00 00 00 09 00 00
020 00 32 00 00 0E B0 00 00-00 02 00 00 0E E2 00 00 ·2··· ····· â··
030 01 1E 00 00 00 00 00 00-00 00 00 00 00 00 00 00
040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
050 00 00 00 00 41 54 54 52-00 00 00 01 00 00 0E E2 ···· ATTR ····· â
060 00 00 00 C8 00 00 00 F4-00 00 00 00 00 00 00 00 ··· È ··· ô
070 00 00 00 00 00 00 00 02-00 00 00 C8 00 00 00 43 ····· È ··· C
080 00 00 15 63 6F 6D 2E 61-70 70 6C 65 2E 71 75 61 ··· com.apple.qua
090 72 61 6E 74 69 6E 65 00-00 00 01 0B 00 00 00 B1 rantine ····· ±
0a0 00 00 25 63 6F 6D 2E 61-70 70 6C 65 2E 6D 65 74 ··%com.apple.met
0b0 61 64 61 74 61 3A 6B 4D-44 49 74 65 6D 57 68 65 adata:kMDItemWhe
0c0 72 65 46 72 6F 6D 73 00-30 30 38 31 3B 35 32 37 reFroms·0081;527
0d0 36 35 36 61 64 3B 47 6F-6F 67 6C 65 5C 78 32 30 656ad;Google\x20
0e0 43 68 72 6F 6D 65 3B 38-44 35 38 45 45 37 43 2D Chrome;8D58EE7C-
0f0 37 44 46 46 2D 34 31 46-32 2D 38 36 43 44 2D 32 7DFF-41F2-86CD-2
100 37 34 30 37 34 45 31 32-46 34 30 62 70 6C 69 73 74074E12F40bplis
110 74 30 30 A2 01 02 5F 10-5C 68 74 74 70 3A 2F 2F t00·· _ \http://
120 63 64 6E 2E 63 75 74 65-73 74 70 61 77 2E 63 6F cdn.cutestpaw.co
130 6D 2F 77 70 2D 63 6F 6E-74 65 6E 74 2F 75 70 6C m/wp-content/upl
140 6F 61 64 73 2F 32 30 31-32 2F 30 36 2F 6C 2D 42 oads/2012/06/l-B
150 72 69 74 61 69 6E 73 2D-70 6F 6C 69 63 65 2D 64 ritains-police-d
160 6F 67 73 2D 6F 66 2D 74-68 65 2D 66 75 74 75 72 ogs-of-the-futur
170 65 2E 6A 70 67 5F 10 21-68 74 74 70 73 3A 2F 2F e.jpg_·!https://
180 77 77 77 2E 67 6F 6F 67-6C 65 2E 63 6F 6D 2F 62 www.google.com/b
190 6C 61 6E 6B 2E 68 74 6D-6C 08 0B 6A 00 00 00 00 lank.html··j····
1a0 00 00 01 01 00 00 00 00-00 00 00 03 00 00 00 00
```
```



Everyone has seen those pervasive, almost annoying `.DS_Store` files—they seem to be everywhere!

These files are created when the Finder application is used to view a directory. While all over macOS drives, these files are copied along with files to external hard drives, thumb drives, or network drives. These files implement a B-tree format and are used to save the Finder viewing settings. These are settings such as the column used to sort by, icon placement, or perhaps of most interest, the Trash "Put Back" capability.

These files can be written on many types of volumes; however, they do not appear to get written to ExFAT volumes.

Just the existence of these files can indicate if a directory was browsed using the Finder application.

References:
https://wiki.mozilla.org/DS_Store_File_Format
https://github.com/dscho/dsstore
https://github.com/nicoleibrahim/DSStoreParser

Trash: ~/.Trash

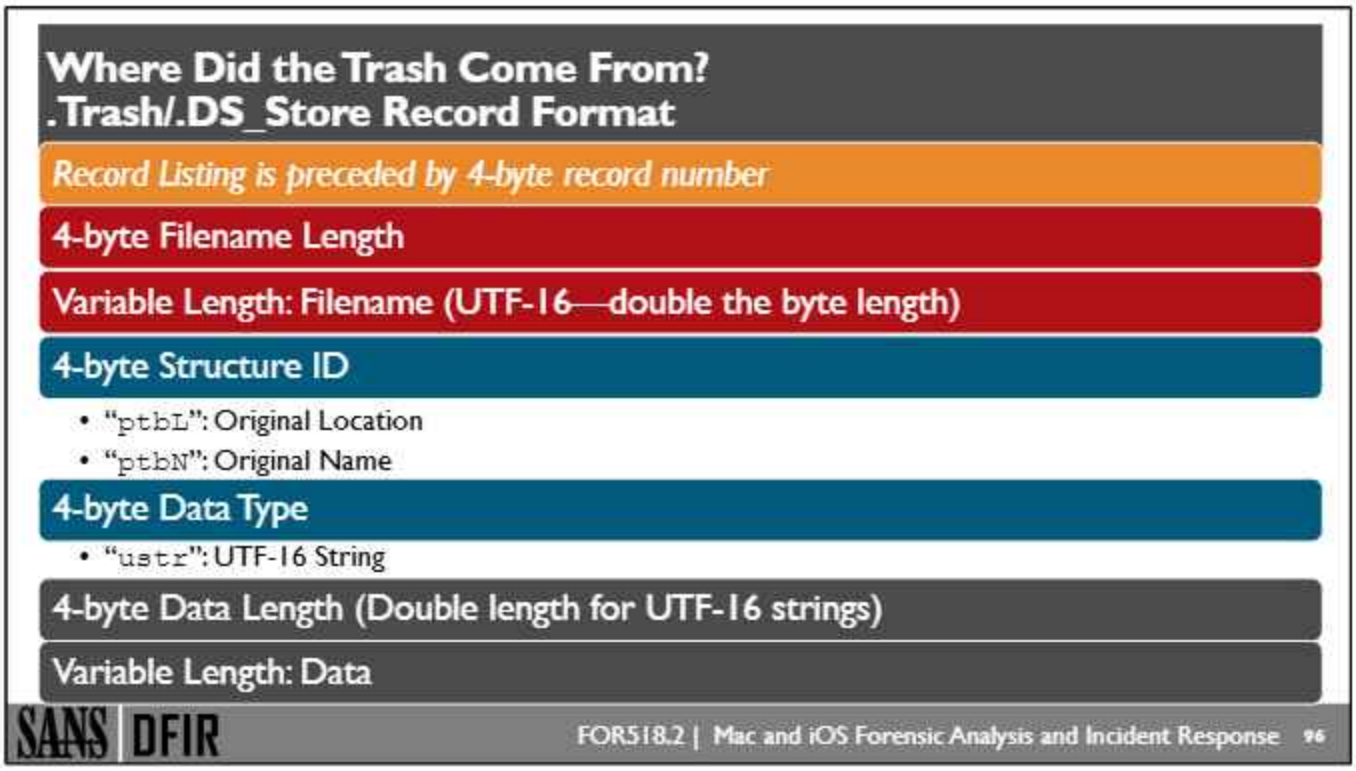
```
janedoe@MBP-M1 ~ % ls -la
total 16
drwxr-xr-x+ 14 janedoe  staff   448 Mar  7 09:45 .
drwxr-xr-x   6 root     admin   192 Mar  7 09:33 ..
-r--------   1 janedoe  staff     7 Mar  7 09:44 .CFUserTextEncoding
drwx------   2 janedoe  staff    64 Mar  7 09:45 .Trash
-rw-------   1 janedoe  staff    47 Mar  7 09:40 .zsh_history
drwx------   6 janedoe  staff   192 Mar  7 09:45 .zsh_sessions
drwx------+  3 janedoe  staff    96 Mar  7 09:33 Desktop
drwx------+  3 janedoe  staff    96 Mar  7 09:33 Documents
drwx------+  3 janedoe  staff    96 Mar  7 09:33 Downloads
drwx------@ 60 janedoe  staff  1920 Mar  7 09:47 Library
drwx------   3 janedoe  staff    96 Mar  7 09:33 Movies
drwx------+  3 janedoe  staff    96 Mar  7 09:33 Music
drwx------+  3 janedoe  staff    96 Mar  7 09:33 Pictures
drwxr-xr-x+  4 janedoe  staff   128 Mar  7 09:33 Public
```


Each user has their own hidden `.Trash` directory. This is located in the root of their home directory.

The `.Trash` contains "deleted" items. Some trashed files can be restored using the "Put Back" option on the right-click context menu.

Some files, usually those that are considered Safari "Safe" files, are sent directly to trash as they are automatically unzipped into the `~/Downloads` folder.

If the file has a "Put Back" option, the data can be found in the hidden `.DS_Store` file in the `.Trash` directory.



Each record in the `.DS_store` file can be identified by a UTF-16 filename. Just prior to this filename is a 4-byte filename length. These are followed by a 4-byte Structure ID and 4-byte Data Type. Some of these Structure IDs and Data Types have been documented in the reference pages below. The final part of the record is the data size and data—this can be a variable size.

Each deleted file will have:

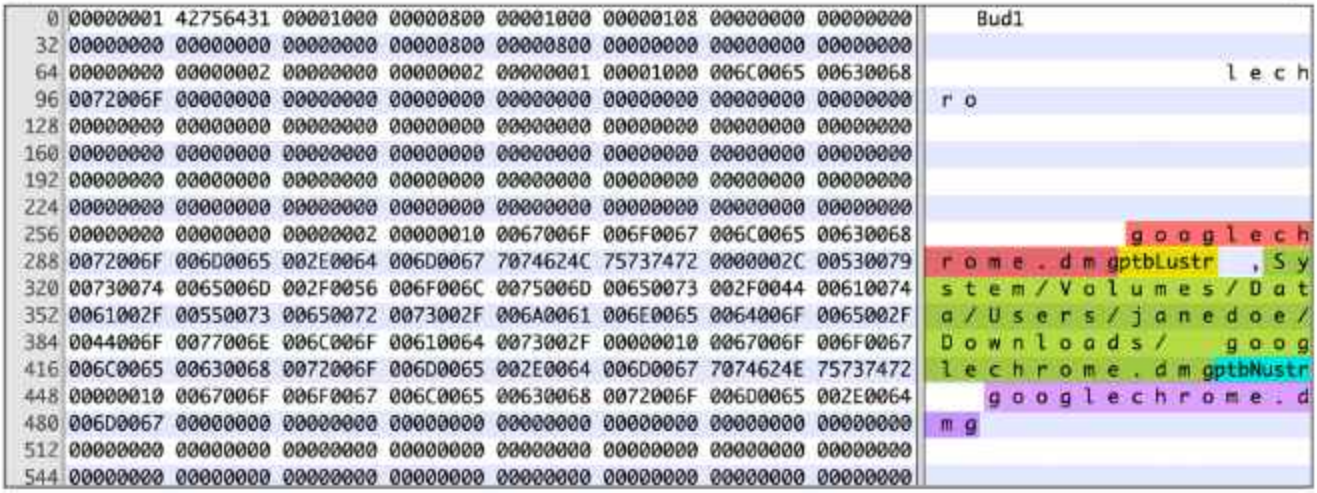
- Filename
- Original File Path

These records are preceded by a 4-byte record number. This will determine how many records are in this file.

There are many more Structure IDs and Data Types that can be seen in .DS_store files, and some have been documented here: https://metacpan.org/pod/distribution/Mac-Finder-DSStore/DSStoreFormat.pod.

References:
https://wiki.mozilla.org/DS_Store_File_Format
https://github.com/dscho/dsstore

Trash .DS_Store Example

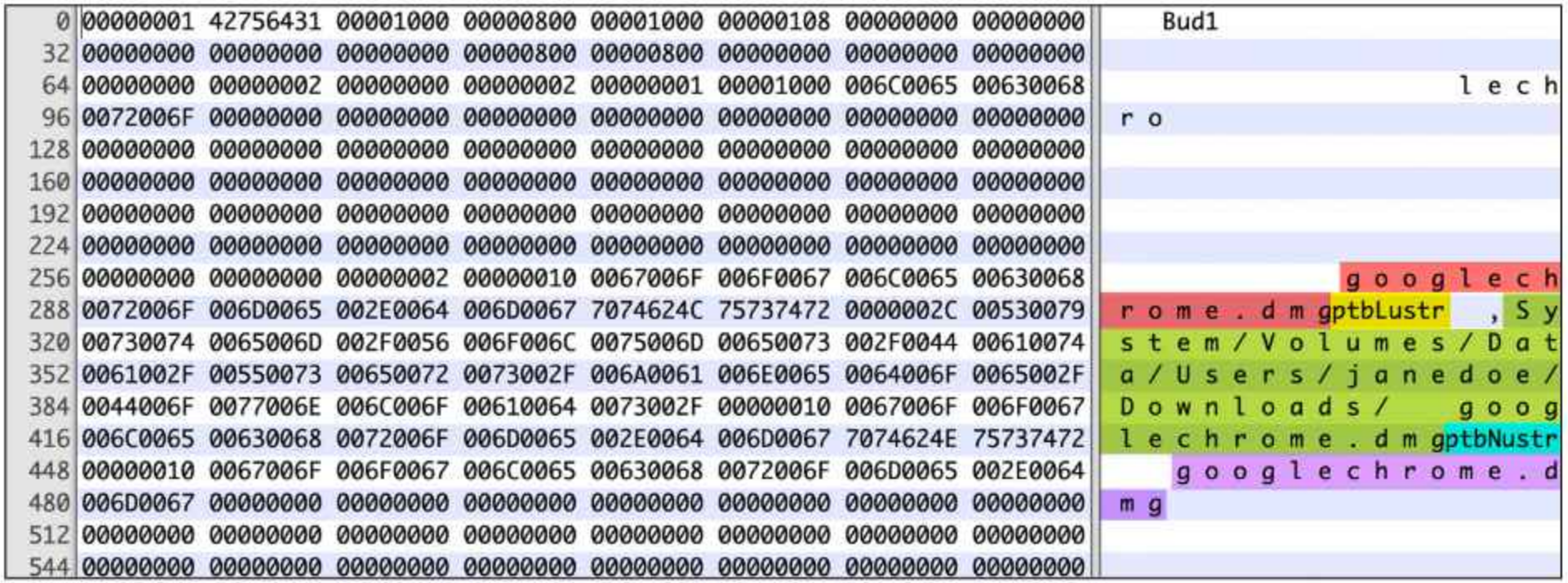


0	00000001	42756431	00001000	00000800	00001000	00000108	00000000	00000000	Bud1
32	00000000	00000000	00000000	00000800	00000800	00000000	00000000	00000000	
64	00000000	00000002	00000000	00000002	00000001	00001000	006C0065	00630068	l e c h
96	0072006F	00000000	00000000	00000000	00000000	00000000	00000000	00000000	r o
128	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
160	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
192	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
224	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
256	00000000	00000000	00000002	00000010	0067006F	006F0067	006C0065	00630068	g o o g l e c h
288	0072006F	006D0065	002E0064	006D0067	7074624C	75737472	0000002C	00530079	r o m e . d m gptbLustr , S y
320	00730074	0065006D	002F0056	006F006C	0075006D	00650073	002F0044	00610074	s t e m / V o l u m e s / D a t
352	0061002F	00550073	00650072	0073002F	006A0061	006E0065	0064006F	0065002F	a / U s e r s / j a n e d o e /
384	0044006F	0077006E	006C006F	00610064	0073002F	00000010	0067006F	006F0067	D o w n l o a d s / g o o g
416	006C0065	00630068	0072006F	006D0065	002E0064	006D0067	7074624E	75737472	l e c h r o m e . d m gptbNustr
448	00000010	0067006F	006F0067	006C0065	00630068	0072006F	006D0065	002E0064	g o o g l e c h r o m e . d
480	006D0067	00000000	00000000	00000000	00000000	00000000	00000000	00000000	m g
512	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
544	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	

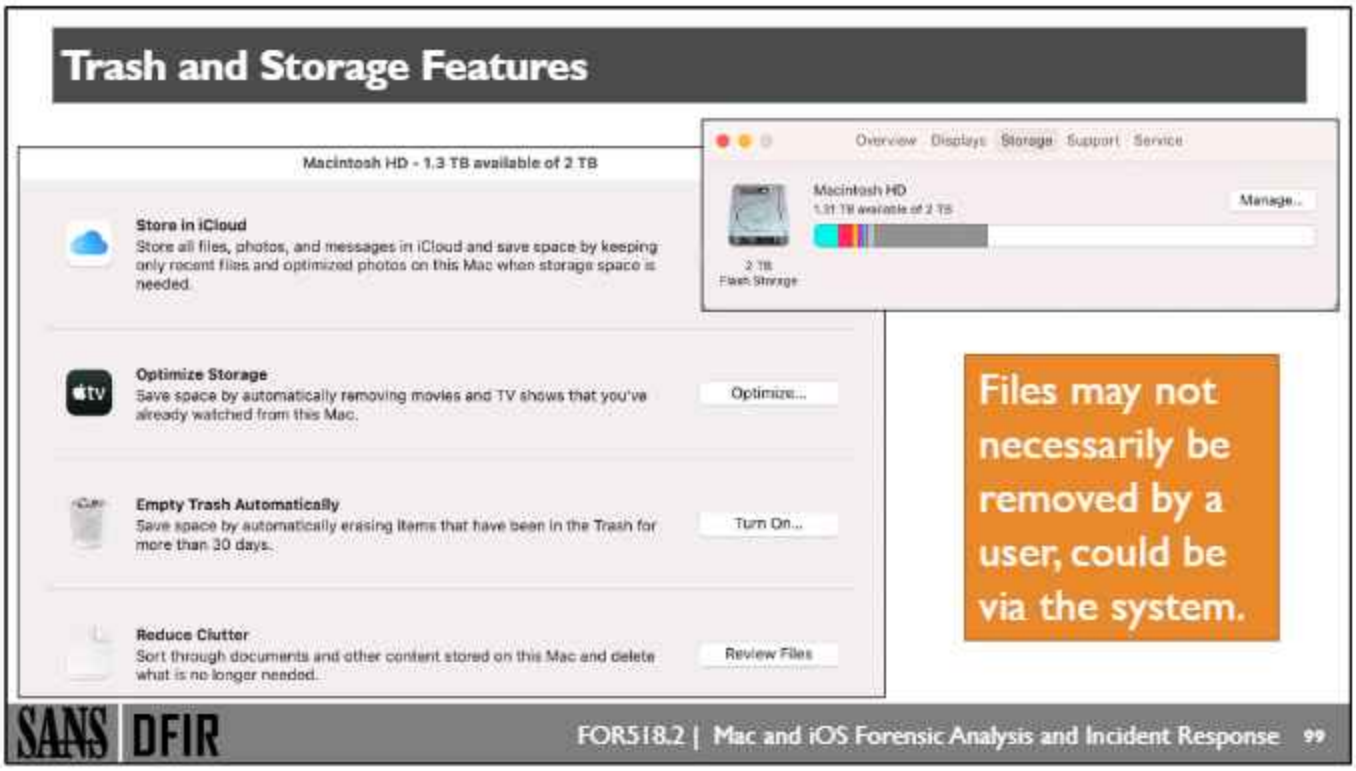
... Data After

Searching for the strings '`ptbLustr`' and '`ptbNustr`', we can find the remnants of the trashed file.

The `APOLLO-master.zip` file will not be found in this file.



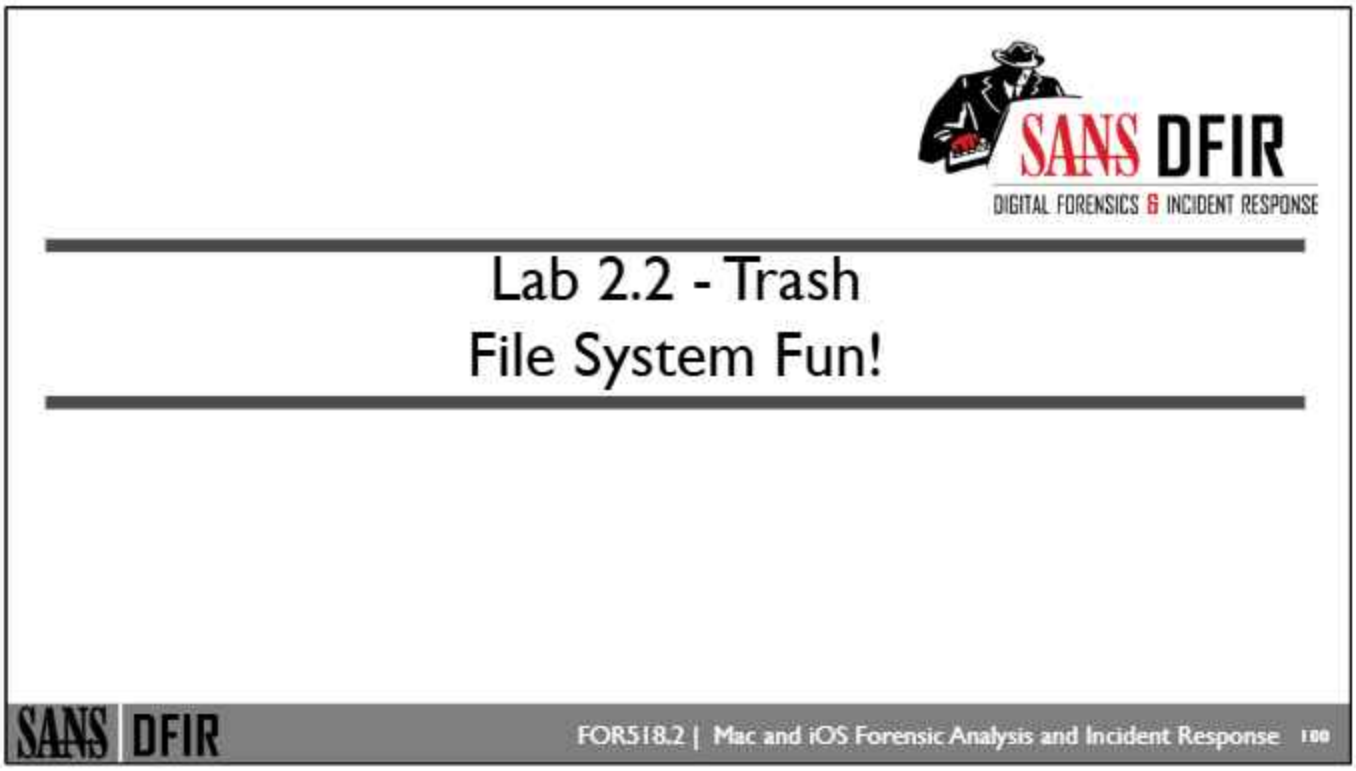
```
0   00000001 42756431 00001000 00000800 00001000 00000108 00000000 00000000    Bud1
32  00000000 00000000 00000000 00000800 00000800 00000000 00000000 00000000
64  00000000 00000002 00000000 00000002 00000001 00001000 006C0065 00630068                                  l e c h
96  0072006F 00000000 00000000 00000000 00000000 00000000 00000000 00000000  r o
128 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
160 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
192 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
224 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
256 00000000 00000000 00000002 00000010 0067006F 006F0067 006C0065 00630068                        g o o g l e c h
288 0072006F 006D0065 002E0064 006D0067 7074624C 75737472 0000002C 00530079  r o m e . d m gptbLustr   , S y
320 00730074 0065006D 002F0056 006F006C 0075006D 00650073 002F0044 00610074  s t e m / V o l u m e s / D a t
352 0061002F 00550073 00650072 0073002F 006A0061 006E0065 0064006F 0065002F  a / U s e r s / j a n e d o e /
384 0044006F 0077006E 006C006F 00610064 0073002F 00000010 0067006F 006F0067  D o w n l o a d s /     g o o g
416 006C0065 00630068 0072006F 006D0065 002E0064 006D0067 7074624E 75737472  l e c h r o m e . d m gptbNustr
448 00000010 0067006F 006F0067 006C0065 00630068 0072006F 006D0065 002E0064      g o o g l e c h r o m e . d
480 006D0067 00000000 00000000 00000000 00000000 00000000 00000000 00000000  m g
512 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
544 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```



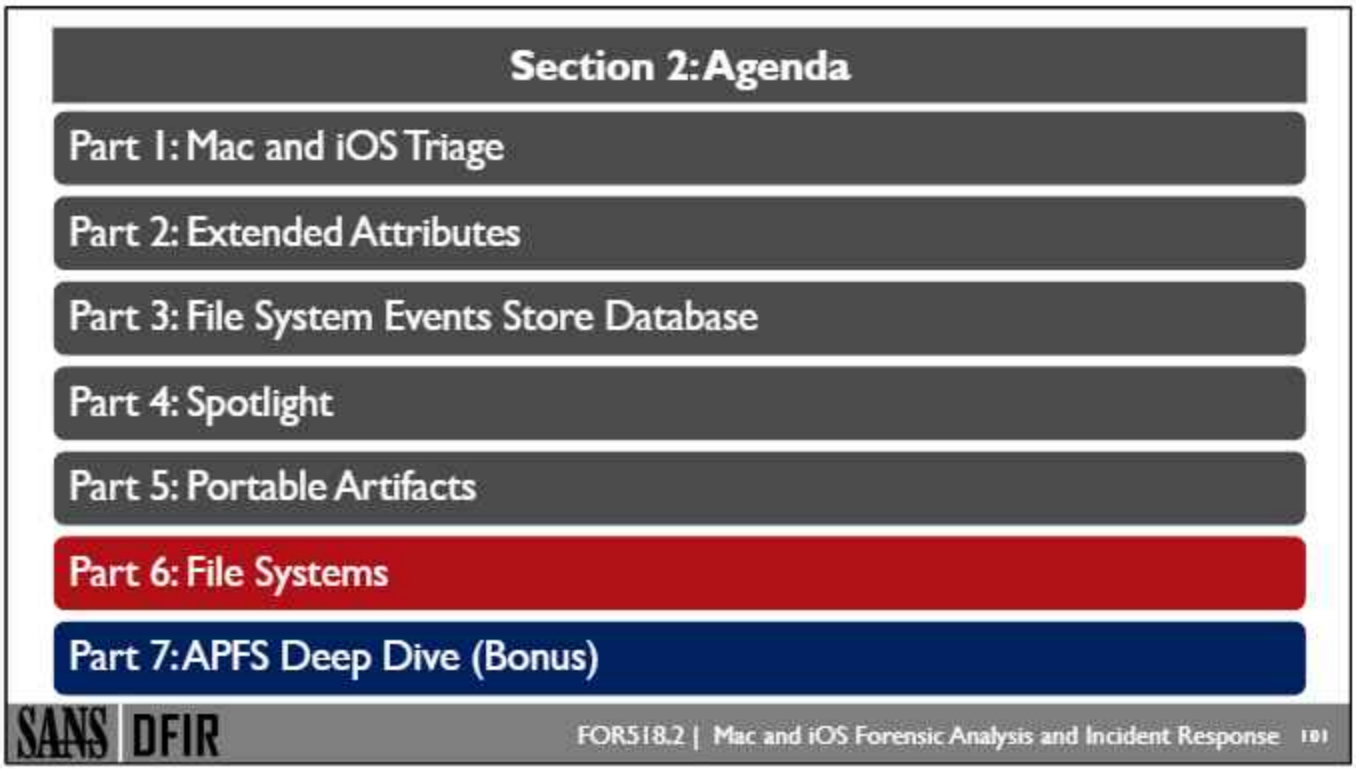
Introduced in Sierra (10.12), macOS now wants to help the user get the most out of the storage space on their systems. The user is reminded of these features when they look at the "`About This Mac | Storage`" menus, as shown in the upper-right screenshot.

A new option for the Trash is to remove items after 30 days, whereas before it was stored in the Trash until the user emptied it.

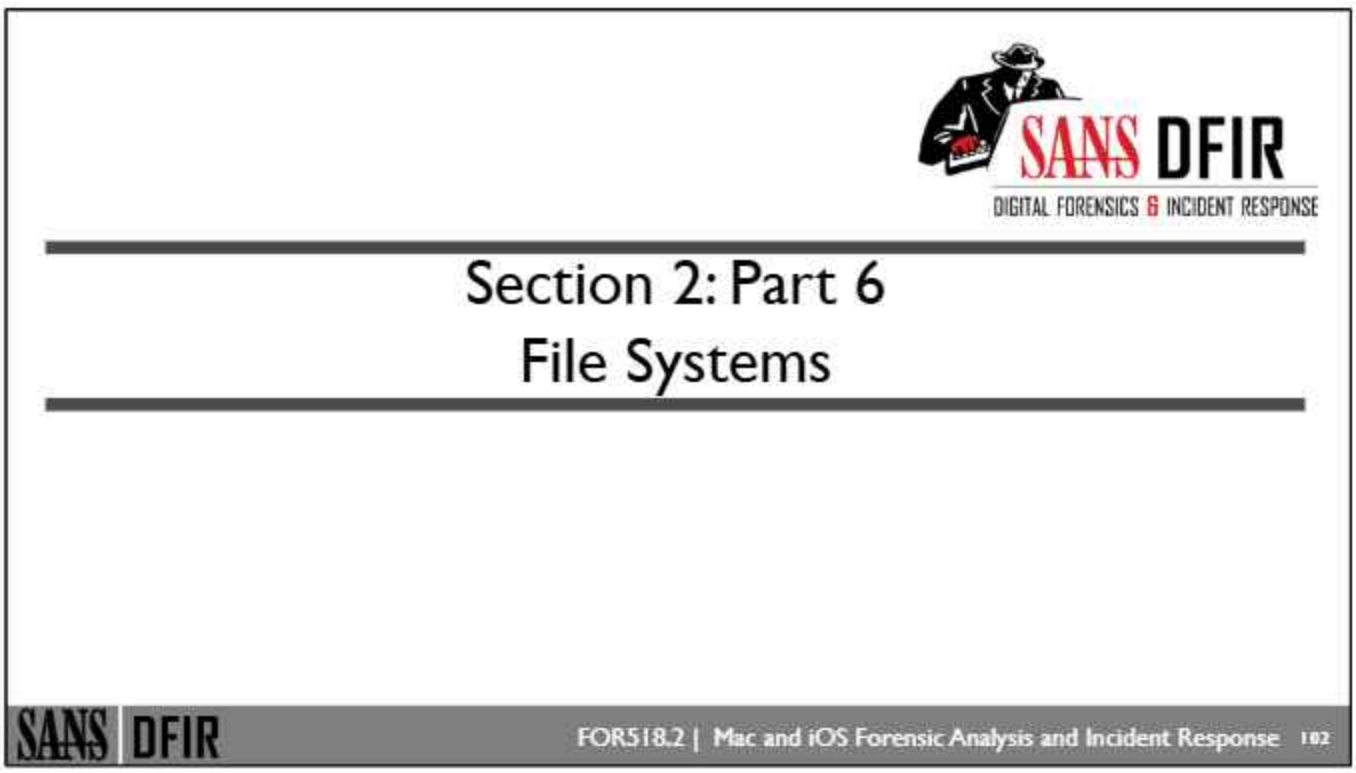
Other functions to be aware of are the "Reduce Clutter" and "Optimize Storage" features. "Reduce Clutter" aims to help the user remove files that have not been used in a while or are very large. The "Review Files" button brings up the window below on the right, which the user can use to delete these items. The "Optimize Storage" feature allows the user to control whether movies are removed after watching and if Mail attachments are downloaded, as shown below on the left.



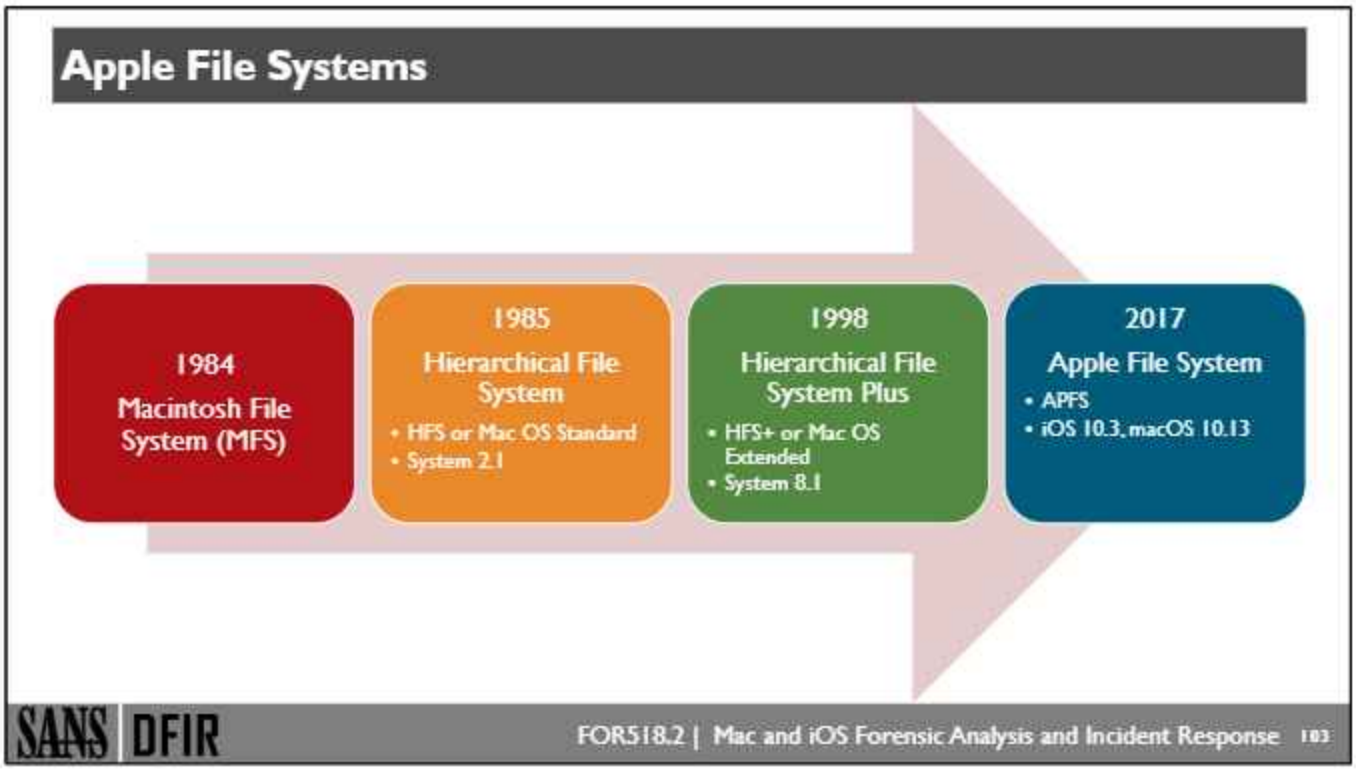
This page intentionally left blank.



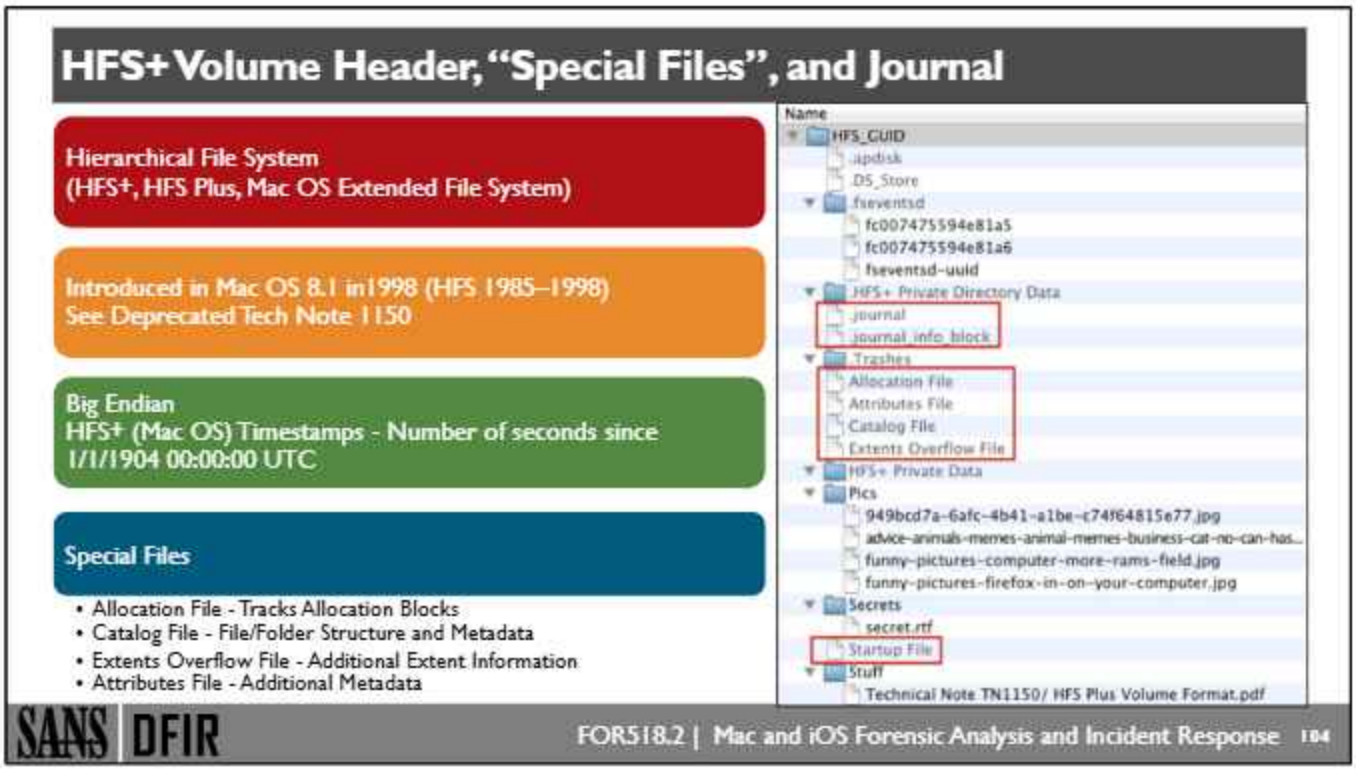
This page intentionally left blank.



This page intentionally left blank.



A major change in recent versions of the operating systems is the change of file system. This will be sure to give us some big analysis challenges!



The Mac OS Extended File System is more widely known as Hierarchical File System or HFS Plus (HFS+). HFS+ was introduced when HFS (Mac OS Standard File System) was limiting the size of the installation media. HFS+ is able to install on larger volumes and have longer filenames, larger file sizes, and more file metadata.

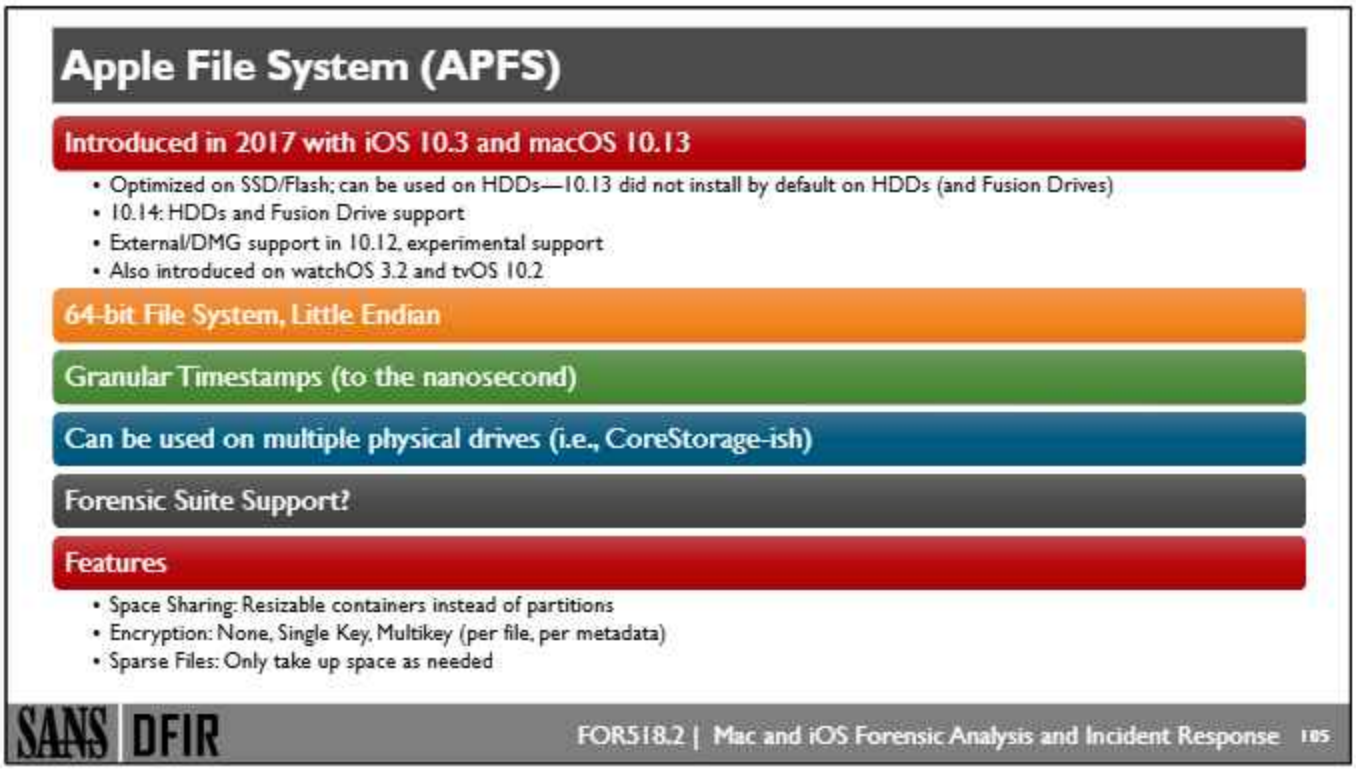
This file system was introduced in Mac OS 8.1 long before Mac OS X/macOS.

HFS+ is comprised of five "special files" and the HFS+ volume header.

The special files are:

- Allocation File
- Catalog File
- Extents Overflow File
- Attributes File
- Startup File

References:
Apple Tech Note 1150: Available at http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html
The Sleuth Kit Source: Available at https://github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h
Mac OS X Internals: A Systems Approach by Amit Singh, Chapter 12
Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin, Chapter 16



References:
https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/APFS_Guide/Introduction/Introduction.html
https://developer.apple.com/videos/play/wwdc2016/701/ (View in Safari.)

APFS Pooled Storage or "Space Sharing" [1]


```
/dev/disk2 (external, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *7.9 GB     disk2
   1:                        EFI EFI                     209.7 MB   disk2s1
   2:                 Apple_APFS Container disk3         7.7 GB     disk2s2

/dev/disk3 (synthesized):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      APFS Container Scheme -                      +7.7 GB     disk3
                                 Physical Store disk2s2
   1:                APFS Volume APFS_VOL1               146.4 MB   disk3s1
   2:                APFS Volume APFS_VOL2               25.5 MB    disk3s2
   3:                APFS Volume APFS_VOL3               10.2 MB    disk3s3
```

```
Sarahs-MBP-6:~ oompa$ df -h /Volumes/APFS_VOL1
Filesystem      Size   Used  Avail Capacity iused                ifree %iused  Mounted on
/dev/disk5s1   7.2Gi  140Mi  7.0Gi     2%      79 9223372036854775728     0%   /Volumes/APFS_VOL1
Sarahs-MBP-6:~ oompa$ df -h /Volumes/APFS_VOL2
Filesystem      Size   Used  Avail Capacity iused                ifree %iused  Mounted on
/dev/disk5s2   7.2Gi   24Mi  7.0Gi     1%     100 9223372036854775707     0%   /Volumes/APFS_VOL2
Sarahs-MBP-6:~ oompa$ df -h /Volumes/APFS_VOL3
Filesystem      Size   Used  Avail Capacity iused                ifree %iused  Mounted on
/dev/disk5s3   7.2Gi  9.7Mi  7.0Gi     1%      88 9223372036854775719     0%   /Volumes/APFS_VOL3
```

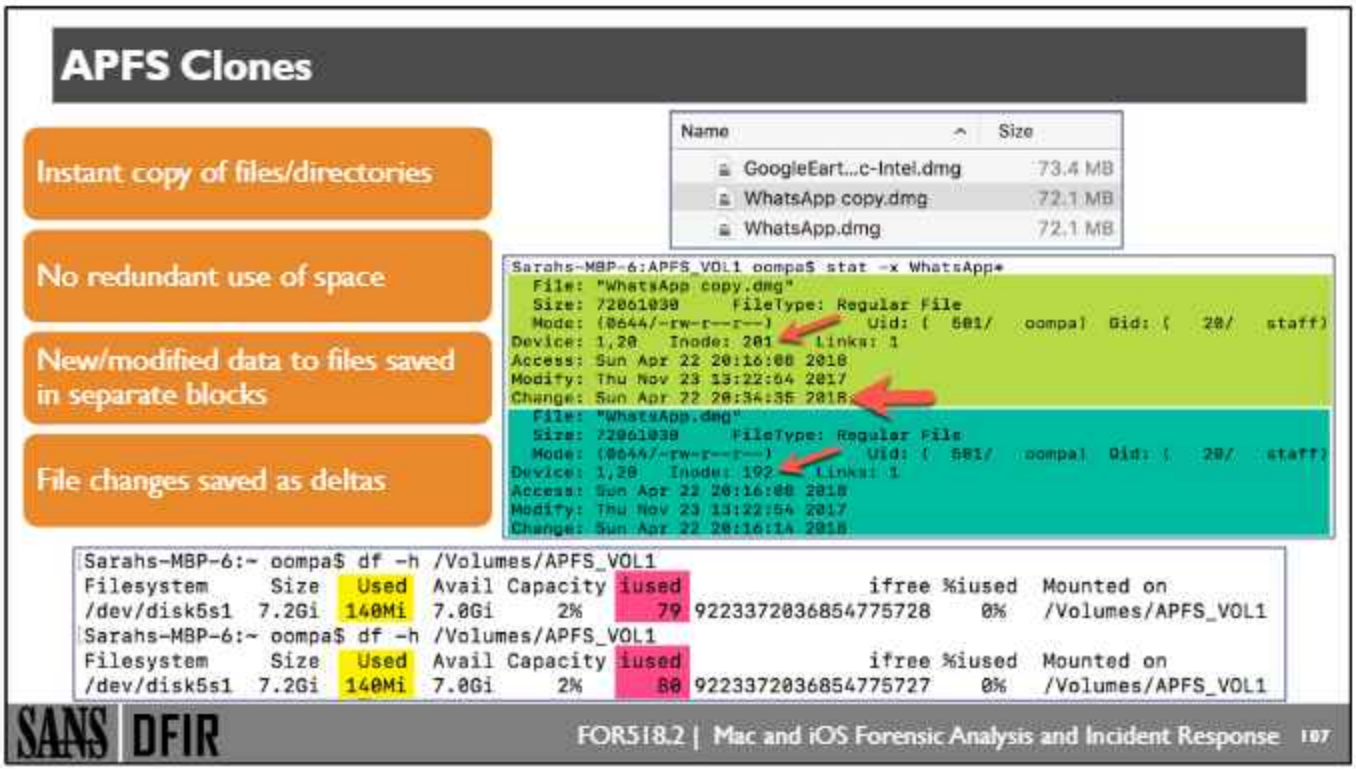

APFS uses "Space Sharing", or pooled storage, meaning that an APFS container may be broken into many APFS volumes that share space. These volumes can grow and shrink as needed using the unallocated space available from what is available on the physical disk(s).

The example above shows a single APFS formatted thumb drive with three APFS volumes: APFS_VOL1, APFS_VOL2, and APFS_VOL3. The files in each of these volumes will take up their needed space as shown in the 'df' command in the screenshot above. The total thumb drive space stays the same, as it is available to all volumes in the APFS container.

The 'diskutil list' output on the right shows each APFS volume as a different "consumed" size. The total for this APFS Container is 8 GB (really, 7.7 GB) but that space is shared among each volume. If the user wanted to add an additional volume, as long as there is space available, it would work.

Each volume can be encrypted separately if the user chooses to do so. If this were the case, the "FileVault:" would show a "Yes" instead of the "No" shown above.

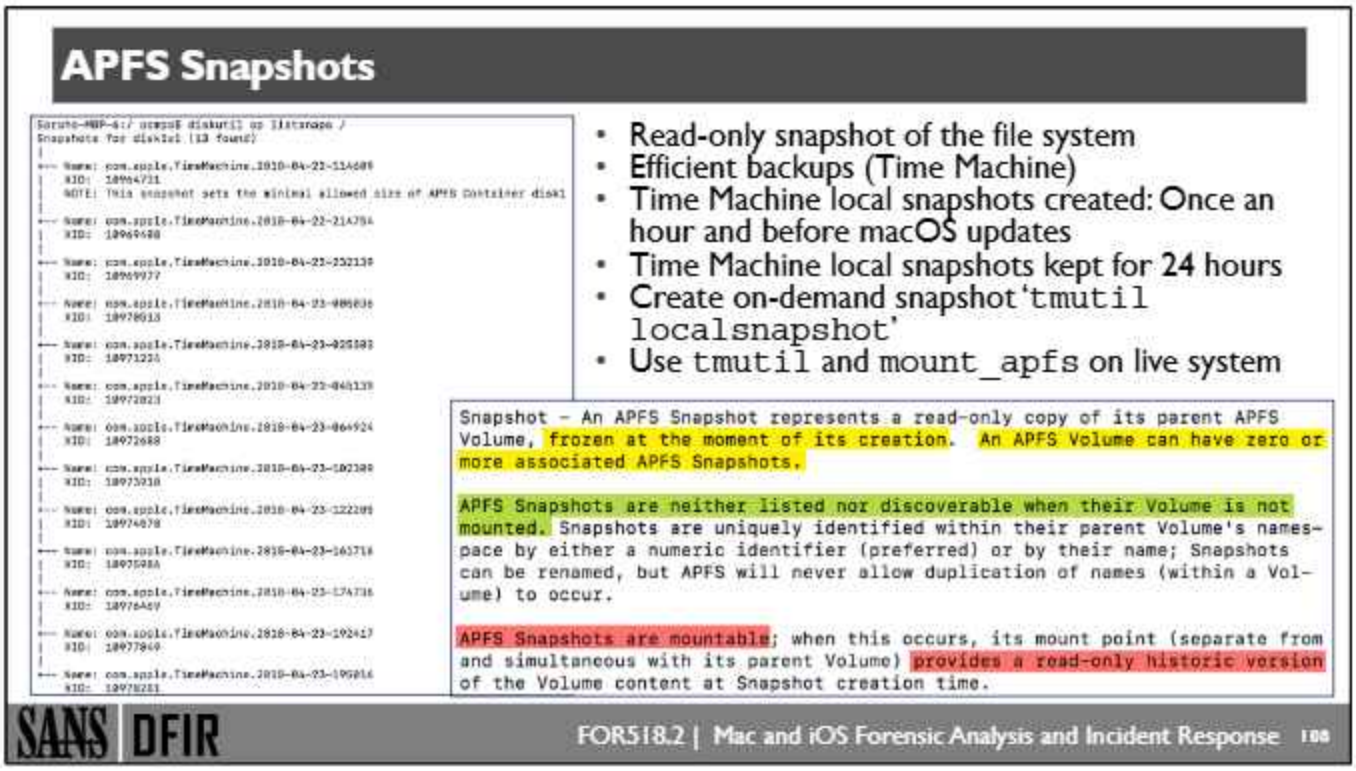
```
Sarahs-MBP-6:~ oompa$ diskutil ap list disk3
|
+-- Container disk3 3B7F97D2-B4FA-46A1-9DD6-04269C6124FD
    ====================================================
    APFS Container Reference:     disk3
    Size (Capacity Ceiling):      7709089792 B (7.7 GB)
    Minimum Size:                 228675584 B (228.7 MB)
    Capacity In Use By Volumes:   207392768 B (207.4 MB) (2.7% used)
    Capacity Not Allocated:       7501697024 B (7.5 GB) (97.3% free)
    |
    +-< Physical Store disk2s2 D9182990-2C76-4120-B5B3-E8CEE9F83EE4
    |   ----------------------------------------------------------
    |   APFS Physical Store Disk:   disk2s2
    |   Size:                       7709089376 B (7.7 GB)
    |
    +-> Volume disk3s1 77BBD723-B592-A5B9-A497-B8AB1ECB8A4A
    |   ----------------------------------------------------------
    |   APFS Volume Disk (Role):   disk3s1 (No specific role)
    |   Name:                      APFS_VOL1 (Case-insensitive)
    |   Mount Point:               /Volumes/APFS_VOL1
    |   Capacity Consumed:         146432000 B (146.4 MB)
    |   FileVault:                 No
    |
    +-> Volume disk3s2 B2551186-BA61-42A8-B805-C8974147E8D6
    |   ----------------------------------------------------------
    |   APFS Volume Disk (Role):   disk3s2 (No specific role)
    |   Name:                      APFS_VOL2 (Case-insensitive)
    |   Mount Point:               /Volumes/APFS_VOL2
    |   Capacity Consumed:         25513984 B (25.5 MB)
    |   FileVault:                 No
    |
    +-> Volume disk3s3 7C6F51AE-D751-4F27-823A-1BECCE4B89AA
        ----------------------------------------------------------
        APFS Volume Disk (Role):   disk3s3 (No specific role)
        Name:                      APFS_VOL3 (Case-insensitive)
        Mount Point:               /Volumes/APFS_VOL3
        Capacity Consumed:         10186752 B (10.2 MB)
        FileVault:                 No
```



APFS uses clones to create instant copies of files without the need to use redundant blocks on the file system for storage. If the cloned file changes, the changes will be saved in deltas on the file system.

The example above shows the cloned file "WhatsApp copy.dmg" that was created from "WhatsApp.dmg". The 'stat' command output shows that the inode number changed, as did the file system metadata (timestamps). The file size listed is also the same, as the file did not change—it was just a copy/clone.

If the 'df' command is used on this volume after the cloned file was created, it will show one more inode used, but the space on the volume stays the same—no additional file system blocks were used.



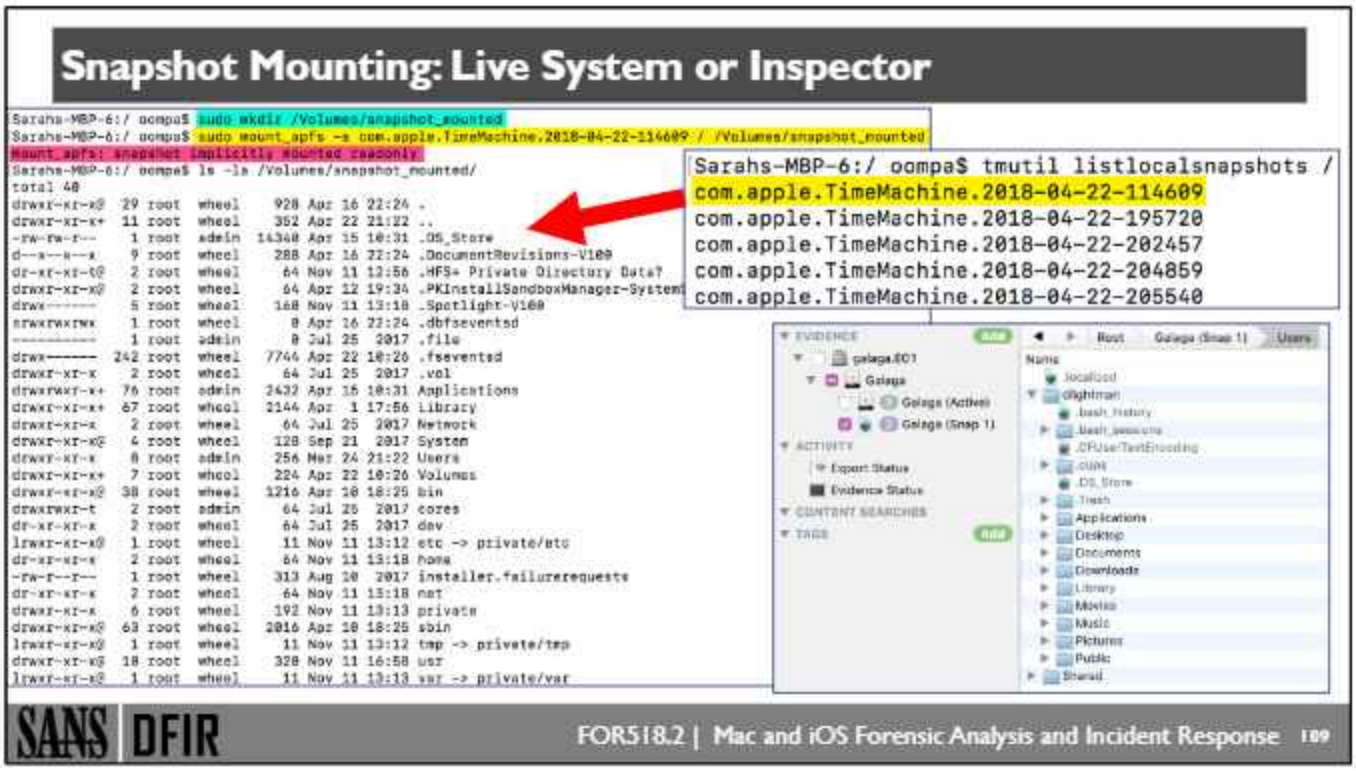
APFS Snapshots are instant copies of file system data. They are currently only used for systems that are using Time Machine. Utilities such as 'tmutil' and 'mount_apfs' can be used to interact with these snapshots.

'`diskutil ap listsnapshots`' (or '`diskutil ap listsnaps`') can be used to view what APFS Snapshots are available to use.

The '`tmutil`' utility can then be used to mount one of these local snapshots to a point where it can be browsed and analyzed as any other mounted volume on the system.

Reference:
About Time Machine local snapshots: https://support.apple.com/en-us/HT204015

```
Sarahs-MBP-6:/ oompa$ tmutil listlocalsnapshots /
com.apple.TimeMachine.2018-04-22-114609
com.apple.TimeMachine.2018-04-22-214754
com.apple.TimeMachine.2018-04-22-232139
com.apple.TimeMachine.2018-04-23-005036
com.apple.TimeMachine.2018-04-23-025303
com.apple.TimeMachine.2018-04-23-045139
com.apple.TimeMachine.2018-04-23-064924
com.apple.TimeMachine.2018-04-23-102309
com.apple.TimeMachine.2018-04-23-122205
com.apple.TimeMachine.2018-04-23-161716
com.apple.TimeMachine.2018-04-23-174736
com.apple.TimeMachine.2018-04-23-192417
com.apple.TimeMachine.2018-04-23-195016
```

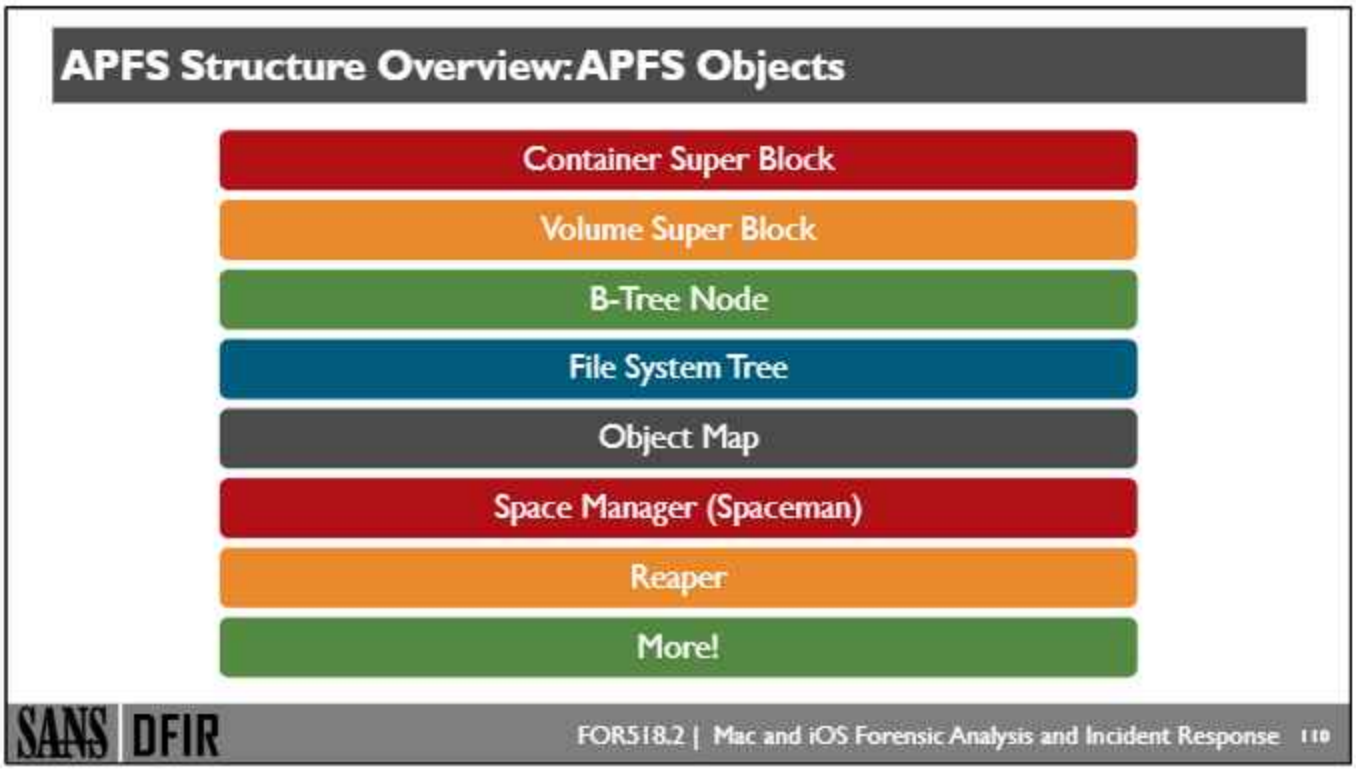


The live APFS snapshots can be mounted using the mount_apfs command just like regular APFS disk images can. The only difference is using the '-s' argument, which specifies the snapshot to mount. This can give the user a historical view of a previous version of the user's system.

The snapshots are mounted in a read-only state on the specified mount point.

Inspector has capability to parse the APFS Snapshots; you'll find this option in the Advanced Parsing area.

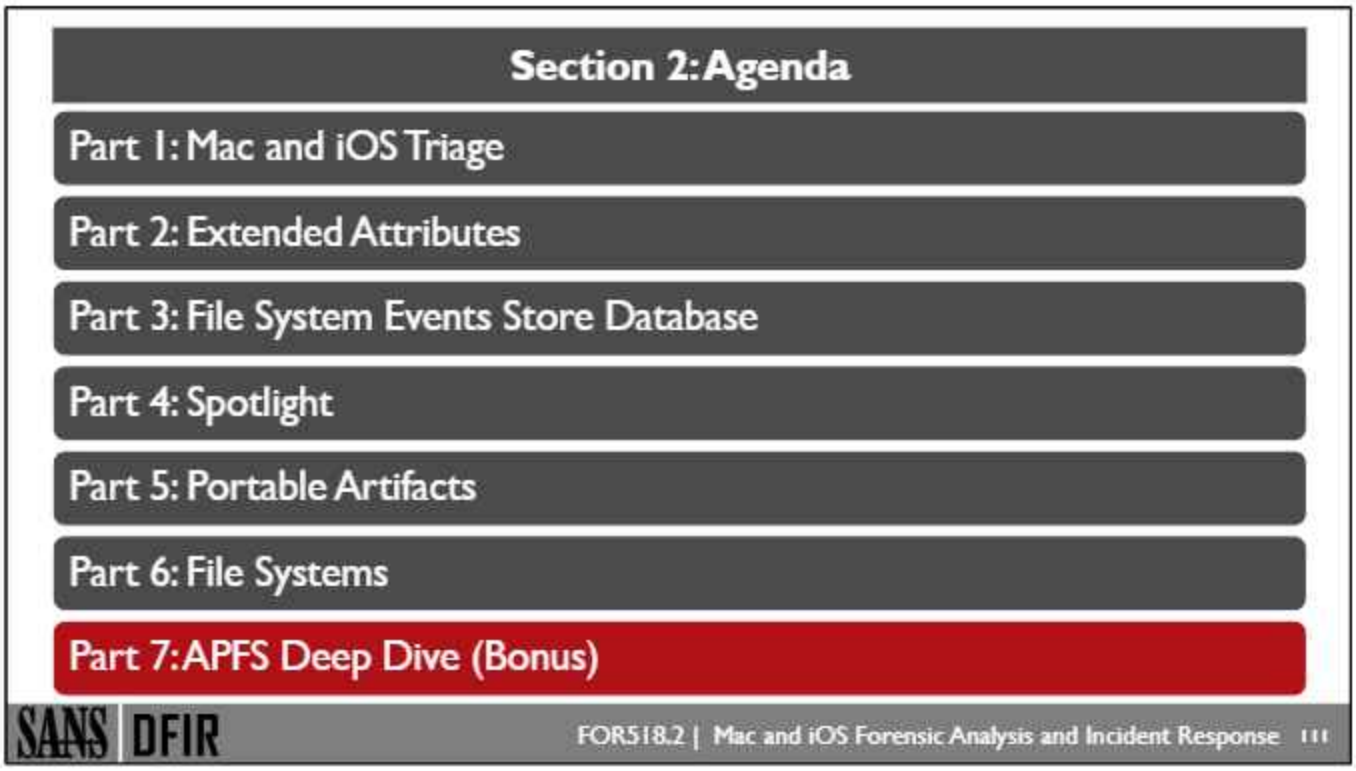
Once parsed, you'll have an additional Volume (Snap 1), depending on how many snapshots were parsed from the APFS volume.



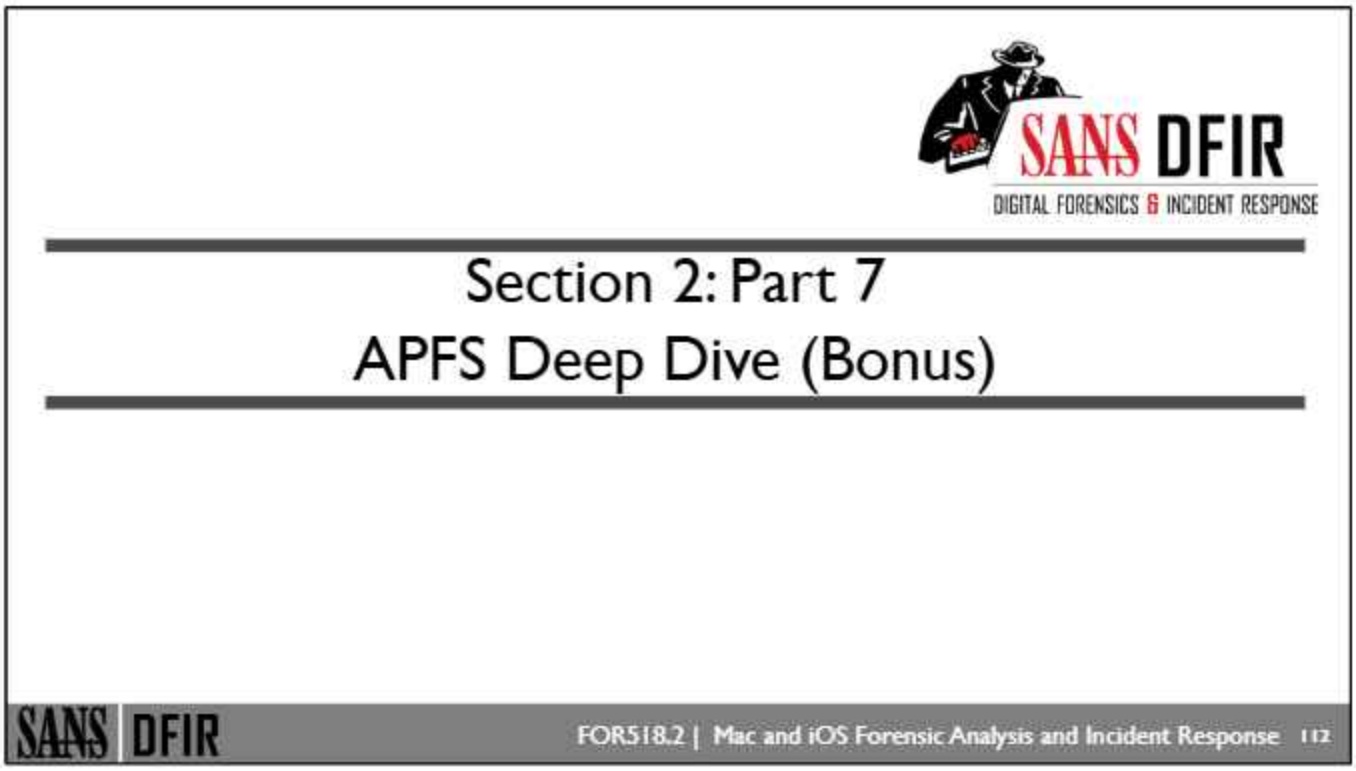
Unlike HFS+, there are no "Special Files" on APFS. All the file system structures are embedded in the file system as objects.

Objects are stored on disk in blocks; a common block size is 4096 bytes. Therefore, you can assume you have a new object every block. The organization of the blocks do not matter, as that is rectified in various file system objects keeping track of where everything is.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf



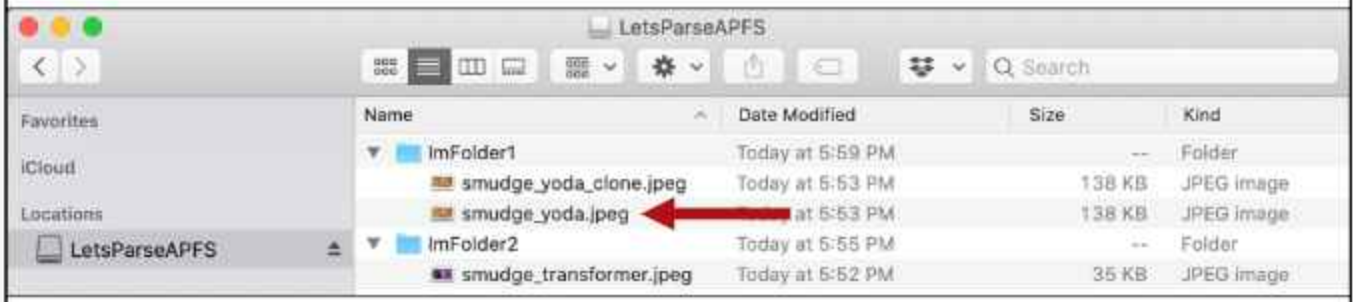
This page intentionally left blank.



This page intentionally left blank.

APFS Structure Overview

- Example APFS Container
  - One Volume: "LetsParseAPFS"
  - Two Directories
  - Cloned File



In the next few slides, we will be diving into the structures associated with the "LetsParseAPFS" DMG file shown above.

This DMG file is simple, yet it shows many of the features you will find in a full file system APFS disk.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Object Header and Types [obj_phys_t]: 32 Bytes

Offset	Size (in bytes)	Field	Notes
0	8	o_cksum	Fletcher 64 Checksum
8	8	o_oid	Object ID
16	8	o_xid	Transaction ID
24	2	o_type.type	**Object Type**
26	2	o_type.flags	Object Flags
28	4	o_subtype	**Object Subtype**

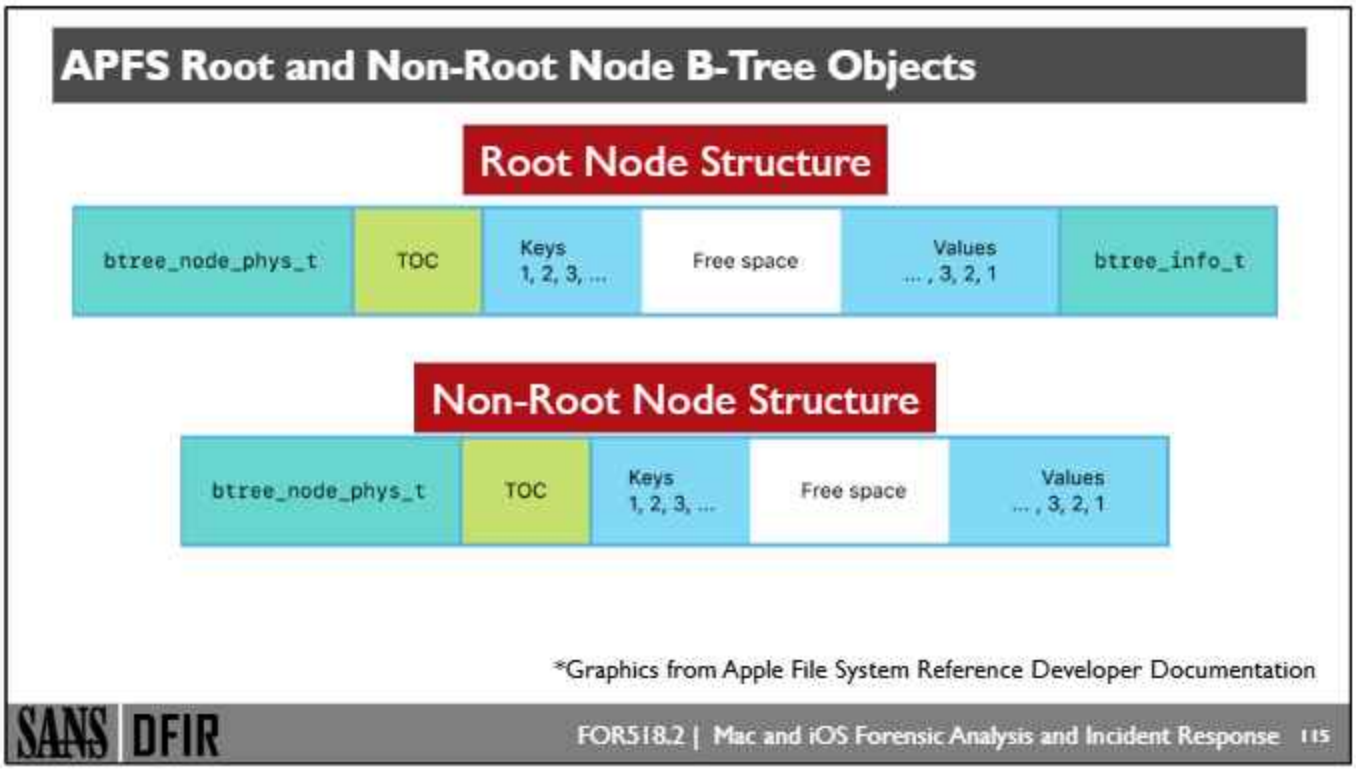
Object Type (Hex)	Object Type (Dec)	Object Type
0x0100	1	**Container Super Block (appendix)**
0x0200	2	B-Tree
0x0300	3	**B-Tree Node**
0x0500	5	Spaceman
0x0B00	11	Object Map (OMAP)
0x0D00	13	**File System (Volume SB, appendix)**
0x1100	17	Reaper

Object Type (Hex)	Object Type (Dec)	Object Subtype
0x0000	0	None
0x0B00	11	Object Map (OMAP)
0x0E00	14	**File System Tree**

Each object in APFS has a 32-byte object header. This header describes what kind of object you are looking at. A few of these objects are listed above.

We will be dissecting a Container Super Block, a Volume Super Block, and a B-Tree Node (with a File System tree).

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf



The file system metadata for each file, directory, extended attribute, etc. is stored in a B-Tree Object, much like that of an HFS+ file system.

The structure that we will look at in this class is a Non-Root Node B-Tree. This B-Tree contains leaf nodes that hold all the file system metadata for each file system entry.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree Node (btree_node_phys_t)

Size (in bytes)	Field	Value & Notes
32	obj_phys_t	Object Header (Object Type = 3, B-Tree Node, Subtype = 14, File System Tree)
2	btn_flags	Flags (Leaf Node)
2	btn_level	Number of Child Levels below this Node
4	btn_nkeys	Number of Keys
2	btn_table_space.off	Offset to Table of Contents (after btree_node_phys_t)
2	btn_table_space.len	Length of Table of Contents
2	btn_freespace.off	Offset Key/Value Free Space
2	btn_freespace.len	Length of Key/Value Free Space
2	btn_key_free_list.off	Offset to Free Key Space
2	btn_key_free_list.len	Length of Free Key Space
2	btn_val_free_list.off	Offset to Free Value Space
2	btn_val_free_list.len	Length of Free Value Space

Each B-Tree Node contains a structure to various pieces of the B-Tree. In this structure, you have how many key entries are stored in this B-Tree, along with where the keys and values for the entries are found in the tree.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Table of Contents (TOC)

TOC Entry (8 bytes)	key_offset (2 bytes)	key_length (2 bytes)	value_offset (2 bytes)	value_length (2 bytes)	Object ID (Inode #)
1	0x0000 = 0	0x1800 = 24	0x1200 = 18	0x1200 = 18	01 private-dir
2	0x1800 = 24	0x1100 = 17	0x2400 = 36	0x1200 = 18	01 root
3	0x2900 = 41	0x0800 = 8	0x9000 = 144	0x6C00 = 108	02
4	49	22	162	18	02
5	71	22	180	18	02
6	93	23	198	18	02
... continues on for number of keys (btn_nkeys) ...					

The Table of Contents (TOC) section of the B-Tree node contains the offsets and lengths for each key/value pair in the tree.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Table of Contents (TOC)—Example

B-Tree Offset in Example	TOC Entry	key_offset (2 bytes)	key_length (2 bytes)	value_offset (2 bytes)	value_length (2 bytes)	Object ID (Inode #)
*... continues on for number of keys (**btn_nkeys**) ...*						
264	27	555	8	1405	168	**0x14 = 20**
272	28	563	36	1425	20	**0x14 = 20**
280	29	599	31	1486	61	**0x14 = 20**
288	30	630	28	1516	30	**0x14 = 20**
296	31	658	8	1520	4	**0x14 = 20**
304	32	666	16	1544	24	**0x14 = 20**
*... continues on for number of keys (**btn_nkeys**) ...*						

The Table of Contents (TOC) section of the B-Tree node contains the offsets and lengths for each key/value pair in the tree.

The example above contains the TOC contents for a particular file—Inode 20.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: File System Keys (Inode 0x14 [20])

B-tree Offset	Entry	Offset	Size (in bytes)	Object ID (Inode #, 7 bytes)	Entry Kind (Highest byte)	Entry Type	Value and Notes
995	27	555	8	0x140000000 00000 = 20	0x30	Inode	N/A
1003	28	563	36	0x140000000 00000 = 20	0x40	Xattr	com.apple.lastuseddate#PS [2-byte size before, 1 byte padding after]
1039	29	599	31	0x140000000 00000 = 20	0x40	Xattr	com.apple.quarantine [2-byte size before, 1 byte padding after]
1070	30	630	28	0x140000000 00000 = 20	0x40	Xattr	com.dropbox.attrs [2-byte size before, 1 byte padding after]
1098	31	658	8	0x140000000 00000 = 20	0x60	Data Stream	N/A
1106	32	666	16	0x140000000 00000 = 20	0x80	File Extent	0x0000000000000000

Inode 20 (0x14) has six key/value pairs associated with this file. Each pair is a specific type of data associated with the file. In this example we have an Inode, three extended attributes, a data stream, and a file extent. These are the Keys.

B-Tree Offset = 440 bytes + TOC determined offset within Keys Section.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: File System Values (Inode 0x14 [20])

B-tree Offset (start)	Entry	Offset (from end)	Size (in bytes)	Entry Type	Value and Notes
2691	27	1405	168	Inode	**File Metadata (See Example Slide)**
2671	28	1425	20	Xattr	Flags = 0x0200, Length = 0x1000 Data = 0x28C72C5E0000000096AC181700000000
2610	29	1486	61	Xattr	Flags = 0x0200, Length = 0x3900 Data = "0083;5e2cc6e1;Safari;3F80AE91-AAF8-441F-8030-CFEA67C12747"
2580	30	1516	30	Xattr	Flags = 0x0200, Length = 0x1A00 Data = 0x0A120A1059C45688BCFCFFB4000000000007C9FC108FDC8E8E0F
2576	31	1520	4	Data Stream	0x02000000: 2 Reference Counts (Cloned File)
2552	32	1544	24	File Extent	**File Extent (See Example Slide)**

The values in this key/value pair are stored in the bottom of the B-Tree Node. These values contain the forensic goods that analysts are looking for: The contents of the extended attributes, the file/directory metadata, and file extent details.

B-Tree Offset = 4096 bytes - TOC determined offset within Values Section.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Metadata (Inode 0x14 [20]) [1]

B-tree Offset	Inode Entry Offset	Size (in bytes)	Field	Value and Notes
2691	0	8	parent_id	0x1300000000000000 Parent Inode Number = 19
2699	8	8	private_id	Inode Number = 20 0x1400000000000000
2707	16	8	create_time	0x008A47D31443ED15 = 1579992801000000000 = 2020-01-25 22:53:21 UTC
2715	24	8	mod_time	0x008A47D31443ED15 = 1579992801000000000 = 2020-01-25 22:53:21 UTC
2723	32	8	change_time	0x7B6984933743ED15 = 1579992950254102907 = 2020-01-25 22:55:50.254103 UTC
2731	40	8	access_time	0x00D08ECE2C43ED15 = 1579992904000000000 = 2020-01-25 22:55:04 UTC

The file and directory metadata is stored in the inode key/value pair. The structure contains the same details as we had on HFS+ and other file systems.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Metadata (Inode 0x14 [20]) [2]

B-tree Offset	Inode Entry Offset	Size (in bytes)	Field	Value and Notes
2739	48	8	internal_flags	0x1084000000000000 (Cloned + others)
2747	56	4	nchildren or nlink	0x01000000 = 1 link
2751	60	4	default_protection_class	0x00000000
2755	64	4	write_generation_counter	0x03000000
2759	68	4	bsd_flags	0x00000000
2763	72	4	owner	0xF5010000 = 501
2767	76	4	group	0x14000000 = 20
2771	80	2	mode	0xA481 = 1010010010000001 Byte Flip = 1000 000 110 100 100 1000 = 8 (Regular File) 000 = SetUID, SetGID, Sticky bits 110 = 6 (rw-) User Permissions 100 = 4 (r--) Group Permissions 100 = 4 (r--) Other Permissions
2773	82	2	pad1	0x0000
2775	84	8	pad2	0x0000000000000000

122

The file and directory metadata is stored in the inode key/value pair. The structure contains the same details as we had on HFS+ and other file systems.

File Mode - (See tables 15.11-15.13 in File System Forensic Analysis by Brian Carrier.).

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Metadata (Inode 0x14 [20]) [3]

B-tree Offset	Inode Entry Offset	Size (in bytes)	Field	Value and Notes
2783	92	2	xf_num_exts	Number of Extended Fields = 0x0200 = 2
2785	94	2	xf_used_data	Extended Fields Data Used = 0x4000 = 64 bytes
	96	x_field_t = 8 bytes total	Extended Field: x_type (1 byte), x_flags (1 byte), x_size (2 bytes)	
2787	96	4	0x04 = 4, 0x02 (Do Not Copy), 0x1100 = 17 (File Name)	
2791	100	4	0x08 = 8, 0x20 (System Field), 0x2800 = 40 – Data Stream	
2795	104	{17}	File Name	smudge_yoda.jpeg (w/1 padding bytes 0x00), 17 total bytes
2812	120	{40}	Data Stream (Size: First 8 bytes, Allocated: Next 8 bytes)	0x00000000000000 – 7 unused bytes Size: 0x261C020000000000 = 138278 bytes Allocated: 0x0020020000000000 = 139264

The file and directory metadata is stored in the inode key/value pair. The structure contains the same details as we had on HFS+ and other file systems.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Extent (Inode 0x14 [20]) [1]

File Extent Value: 0x0020020000000000 6900000000000000 0000000000000000	
(Allocated) File Size (8 bytes)	0x0020020000000000 = 139264 bytes
Physical Block Location (8 Bytes): Physical Block Number from start of container (add 5 (20,480) blocks for start of disk) (# * 4096) + 20,480 = start of file location in bytes	0x6900000000000000 = 105 = Physical Block Number (105 * 4096) + 20,480 = 450560 bytes
Crypto_ID (8 Bytes)	Encryption Key: 0x0000000000000000

The Inode File Extent information can be used to find the file in the disk image.

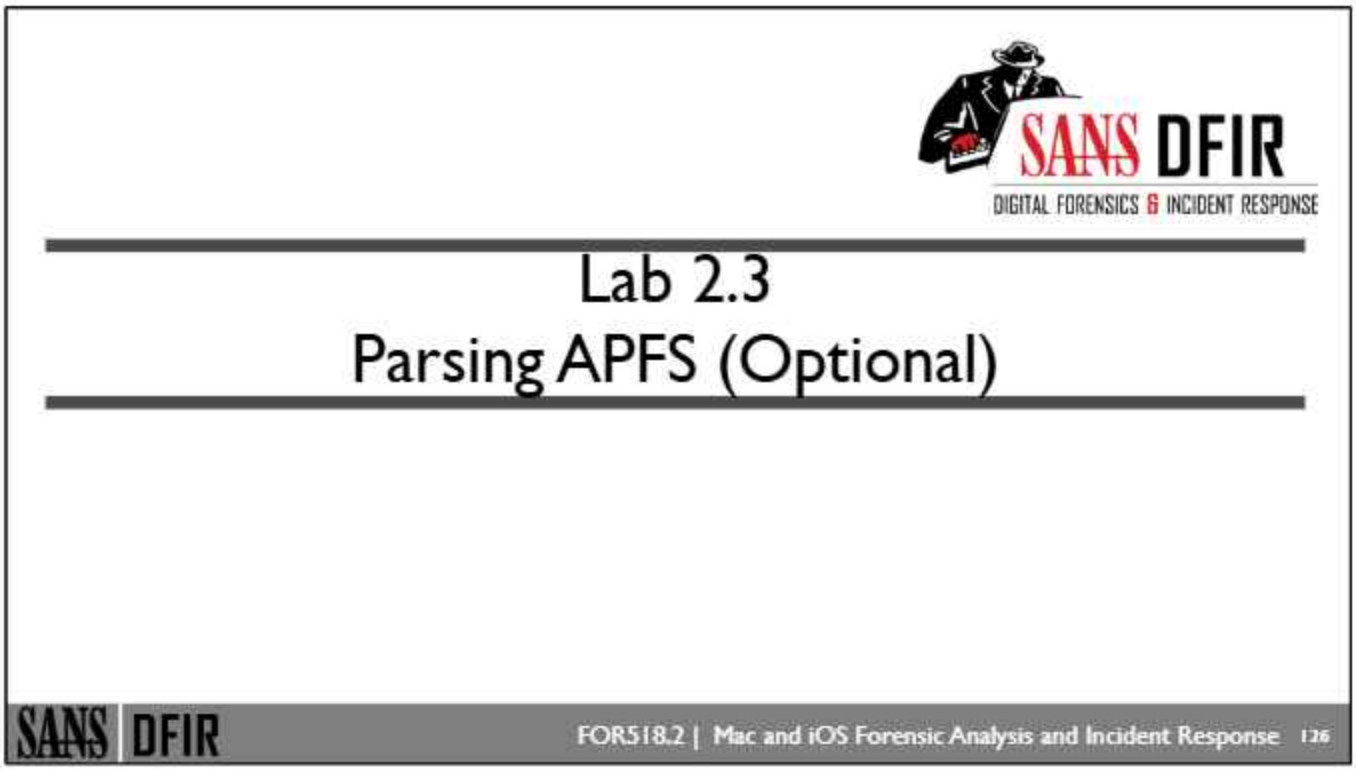
Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Extent (Inode 0x14 [20]) [2]

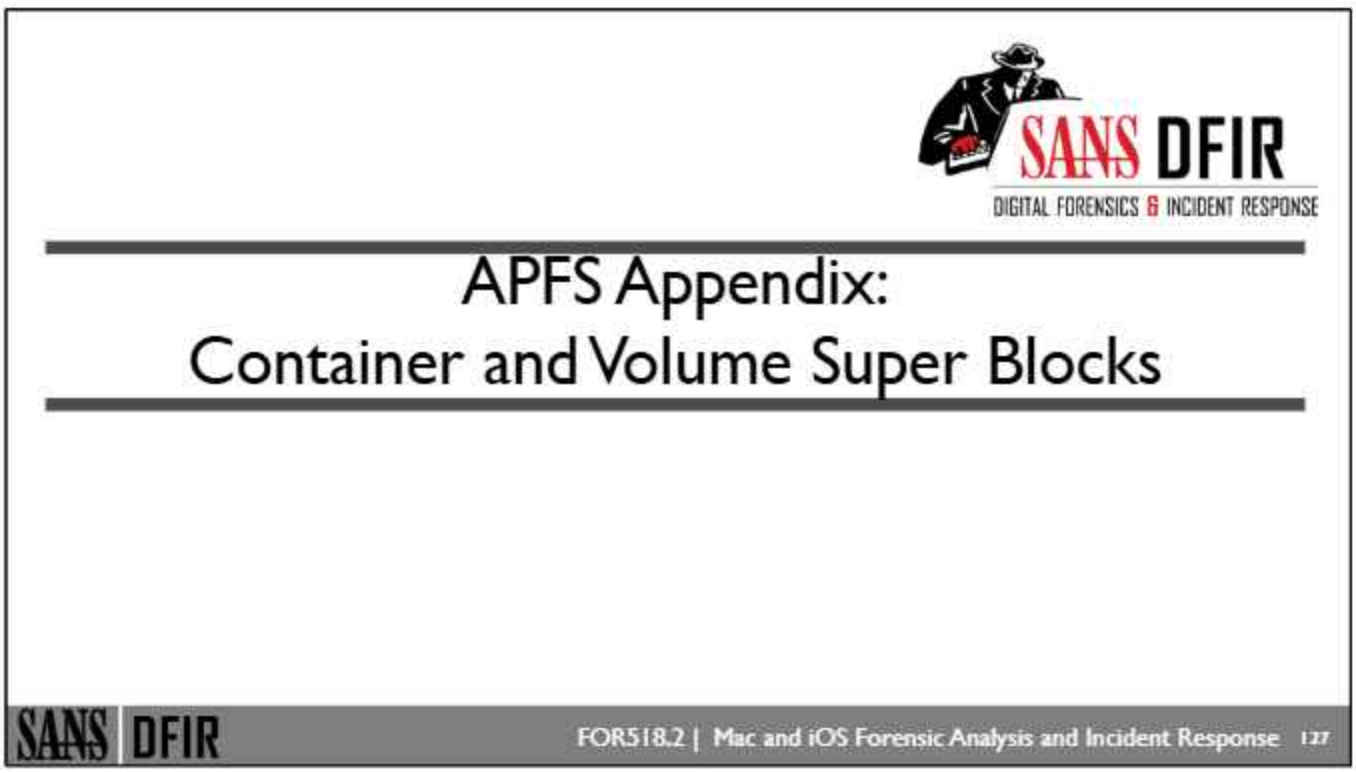
```
dd if=APFS.dmg ibs=1 skip=450560 count=138278 >
~/FOR518/smudge_yoda_extracted.jpeg
```

The Inode File Extent information can be used to find the file in the disk image. The command above is used to extract the file that we have been viewing in this example/demo.

Reference:
Apple File System Reference:
https://developer.apple.com/documentation/foundation/file_system/about_apple_file_system



This page intentionally left blank.



This page intentionally left blank.

APFS Container Super Block (nx_superblock_t) [1]

Size (in bytes)	Field	Notes
32	obj_phys_t	Object Header (Object Type = 1)
4	nx_magic	Container Magic Number: 0x4E585342 = "NSXB"
4	nx_block_size	Block Size (i.e., 4096)
8	nx_block_count	Block Count (Block Count*Block Size = Container Size in Bytes
8	nx_features	Features
8	nx_read_only_ compatible_features	Read-only Compatible Features
8	nx_incompatible_features	Incompatible Features
16	nx_uuid	Container UUID (`diskutil info /dev/disk#`)
8	nx_next_oid	Next Object ID (OID)
8	nx_next_xid	Next Transaction ID (XID)
4	nx_xp_desc_blocks	Blocks used by Checkpoint Descriptor Area
4	nx_xp_data_blocks	Blocks used by Checkpoint Data Area

The structure of the Container Super block is above. This object keeps track of what is going on in an APFS Container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Container Super Block (nx_superblock_t) [2]

Size (in bytes)	Field	Value and Notes
8	nx_xp_desc_base	Base address of Checkpoint Descriptor Area or Physical Object ID
8	nx_xp_data_base	Base address of Checkpoint Data Area or Physical Object ID
4	nx_xp_desc_next	Next Index for Checkpoint Descriptor Area
4	nx_xp_data_next	Next Index for Checkpoint Data Area
4	nx_xp_desc_index	Index for first item in Checkpoint Descriptor Area
4	nx_xp_desc_len	Number of blocks in Checkpoint Descriptor Area Used
4	nx_xp_data_index	Index for first item in Checkpoint Data Area
4	nx_xp_data_len	Number of blocks in Checkpoint Data Area Used
8	nx_spaceman_oid	Space Manager Object ID (OID)
8	nx_omap_oid	Container Object Map Object ID (OID)
8	nx_reaper_oid	Reaper Object ID (OID)
4	nx_test_type	Reserved for Testing
4	nx_max_file_systems	Maximum Number of Volumes in this Container
8	nx_fs_oid[0]	Array of OIDs for Volumes in this Container
...	...	*Continuation of Container Super Block Structure*

The structure of the Container Super block is above. This object keeps track of what is going on in an APFS Container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Volume Super Block (apfs_superblock_t)[1]

Size (in bytes)	Field	Notes
32	obj_phys_t	Object Header (Object Type = 13)
4	apfs_magic "APSB"	Volume Magic Number 0x41505342 = "APSB"
4	apfs_fs_index	Index in Volume Array
8	apfs_features	Features
8	apfs_readonly_compatible_features	Read-only Incompatible Features
8	apfs_incompatible_features	Incompatible Features
8	apfs_unmount_time	Timestamp when volume was last unmounted
8	apfs_fs_reserve_block_count	Block Pre-allocated for Volume (Default is none)
8	apfs_fs_quota_block_count	Maximum Block Allocated (Default is none)
8	apfs_fs_alloc_count	Number of blocks currently allocated

The structure of the Volume Super block is above. This object keeps track of what is going on in an APFS Volume. An APFS container may have more than one APFS volume embedded in it. There may be multiple Volume Super Blocks for each volume in the container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Volume Super Block (apfs_superblock_t)[2]

Size (in bytes)	Field	Notes
2	wrapped_crypto_state_t. wrapped_crypto_state.major_version	Key Encryption Metadata: Major Version
2	wrapped_crypto_state_t. wrapped_crypto_state.minor_version	Key Encryption Metadata: Minor Version
4	wrapped_crypto_state_t. wrapped_crypto_state.cpflags	Key Encryption Metadata: Encryption State Flags
4	wrapped_crypto_state_t. wrapped_crypto_state.persistent_class	Key Encryption Metadata: Protection Class
4	wrapped_crypto_state_t. wrapped_crypto_state.key_os_version	Key Encryption Metadata: Creator OS Version 0x39004313 = 19 C 57: 19C57: Catalina 10.15.2
2	wrapped_crypto_state_t. wrapped_crypto_state.key_revision	Key Encryption Metadata: Key Version
2	wrapped_crypto_state_t. wrapped_crypto_state.key_len	Key Encryption Metadata: Key Size (0 for no Encryption)
0	wrapped_crypto_state_t. wrapped_crypto_state.persistent_key	Key Encryption Metadata: Wrapped Key No Key field is null; see key_len above

131

The structure of the Volume Super block is above. This object keeps track of what is going on in an APFS Volume. An APFS container may have more than one APFS volume embedded in it. There may be multiple Volume Super Blocks for each volume in the container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Volume Super Block (apfs_superblock_t)[3]

Size (in bytes)	Field	Notes
4	apfs_root_tree_oid_type	Type of Root File System Tree = B-Tree
4	apfs_extentref_tree_oid_type	Type of Extent Reference Tree = B-Tree, Physical
4	apfs_snap_meta_tree_oid_type	Type of Snapshot Metadata Tree = B-Tree, Physical
8	apfs_omap_oid	Physical Object ID (OID) of Object Map
8	apfs_root_tree_oid	Virtual Object ID (OID) of Root File System Tree
8	apfs_extentref_tree_oid	Physical Object ID (OID) of Extent Reference Tree
8	apfs_snap_meta_tree_oid	Virtual Object ID (OID) of Snapshot Metadata Tree
8	apfs_revert_to_xid	Transaction ID (XID) that volume will revert to
8	apfs_revert_to_sblock_oid	Virtual Object ID (OID) of Volume Superblock to revert to
8	apfs_next_obj_id	Next Object ID (OID)
8	apfs_num_files	Number of Regular Files
8	apfs_num_directories	Number of Directories
8	apfs_num_symlinks	Number of Symbolic Links
8	apfs_num_other_fsobjects	Number of Other Files
8	apfs_num_snapshots	Number of Snapshots

The structure of the Volume Super block is above. This object keeps track of what is going on in an APFS Volume. An APFS container may have more than one APFS volume embedded in it. There may be multiple Volume Super Blocks for each volume in the container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Volume Super Block (apfs_superblock_t)[4]

Size (in bytes)	Field	Notes
8	apfs_total_blocks_alloced	Blocks Allocated by Volume
8	apfs_total_blocks_freed	Blocks Freed by Volume
16	apfs_vol_uuid	Volume UUID (diskutil info /dev/disk#s# [Volume])
8	apfs_last_mod_time	Last Modified Timestamp
8	apfs_fs_flags	Flags
32	apfs_modified_by_t.formatted_by.id[]	Format Program and Version
8	apfs_modified_by_t.formatted_by. timestamp	Format Timestamp
8	apfs_modified_by_t.formatted_by. last_xid	Format Transaction ID (XID)
32	apfs_modified_by_t.modified_by.id[]	Last Modified Program and Version
8	apfs_modified_by_t.modified_by. timestamp	Last Modified Timestamp
8	apfs_modified_by_t.modified_by. last_xid	Last Modified Transaction ID (XID)
336	apfs_modified_by_t.modified_by[1-7]	Array of apfs_modified_by_t[8]
256	apfs_volname	APFS Volume Name
4	apfs_next_doc_id	Next Document ID
2	apfs_role	APFS Role (None, System, Data, Preboot, VM, Recovery)
2	apfs_reserved	Reserved
8	apfs_root_to_xid	Transaction ID (XID) of Snapshot to Root
8	apfs_er_state_oid	Current State of Encryption/Decryption

133

The structure of the Volume Super block is above. This object keeps track of what is going on in an APFS Volume. An APFS container may have more than one APFS volume embedded in it. There may be multiple Volume Super Blocks for each volume in the container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf