518.3

User Data, System Configuration, and Log Analysis

SANS

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

518.3

User Data, System Configuration, and Log Analysis

SANS

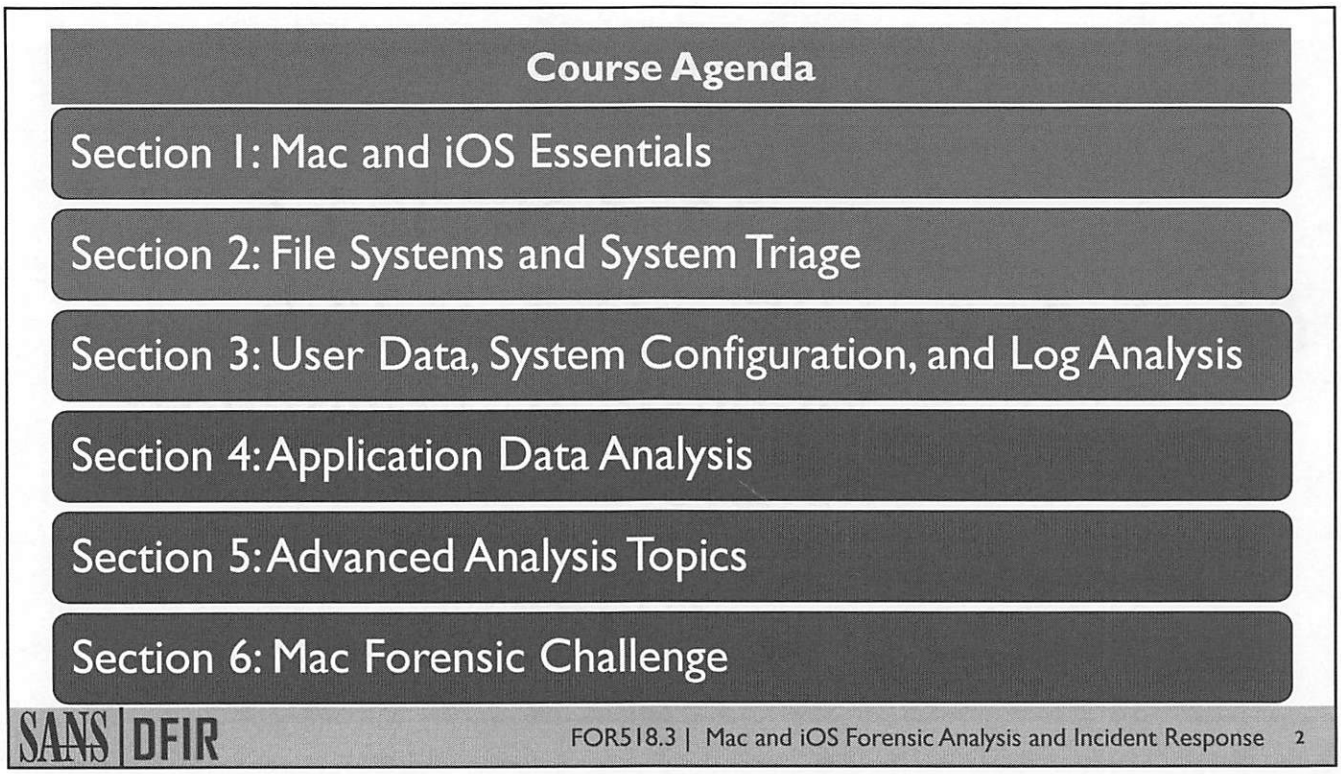
THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

FOR518.3 | Mac and iOS Forensic Analysis and Incident Response

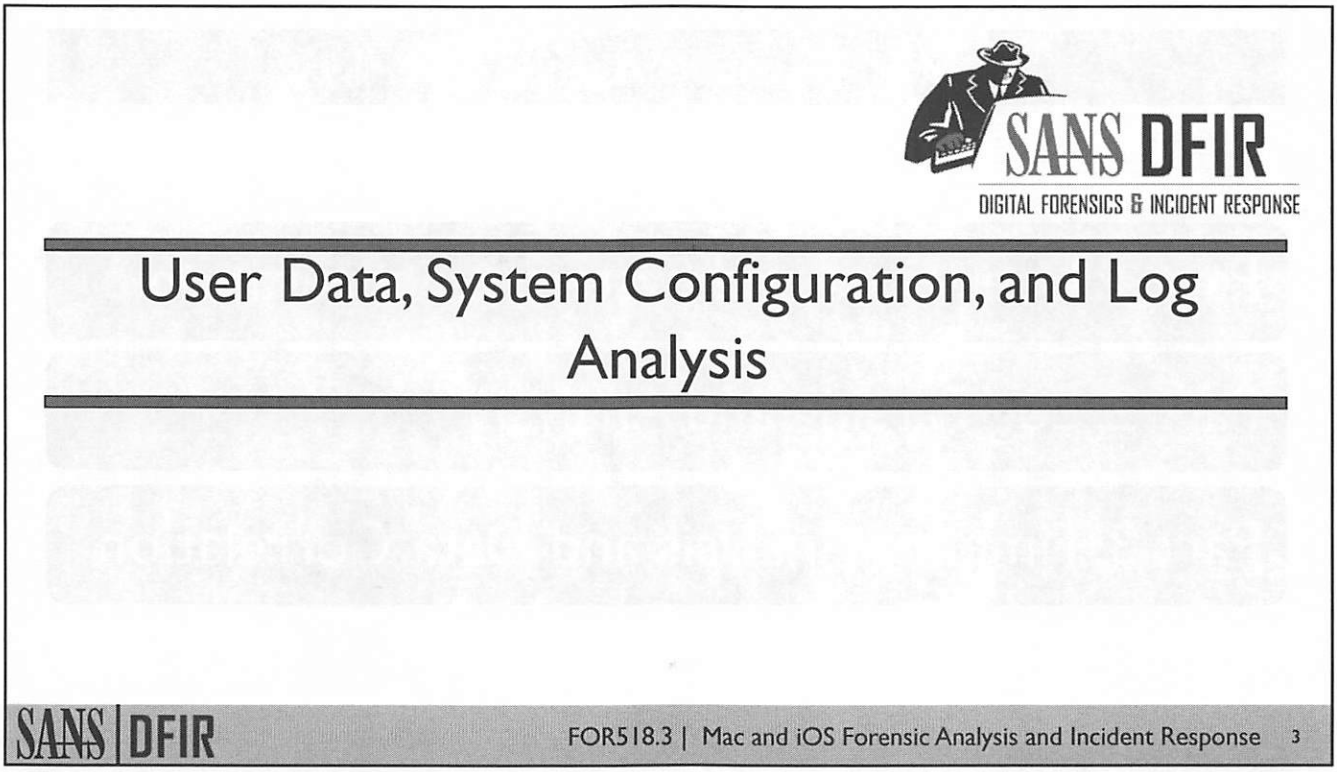
FOR518 Section 3: User Data, System Configuration, and Log Analysis

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

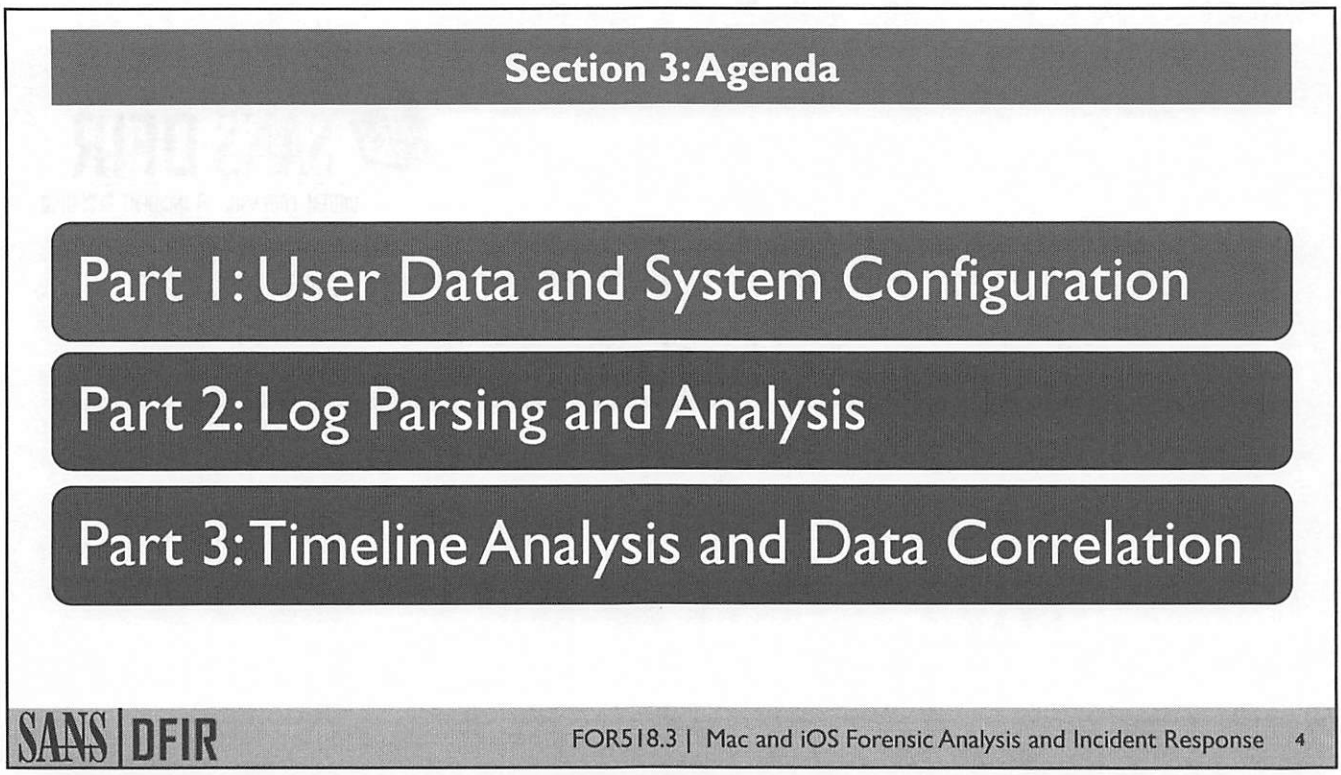
https://digital-forensics.sans.org/
http://twitter.com/sansforensics



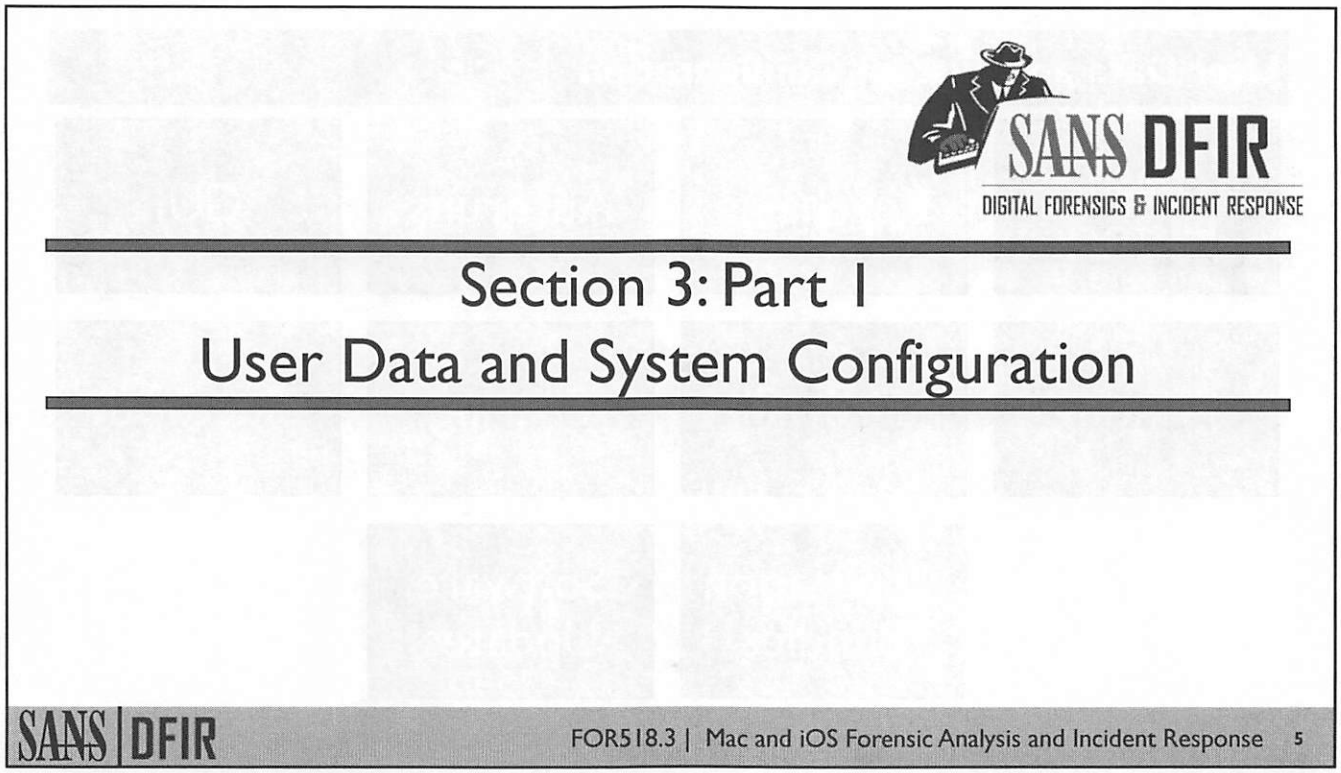
This page intentionally left blank.



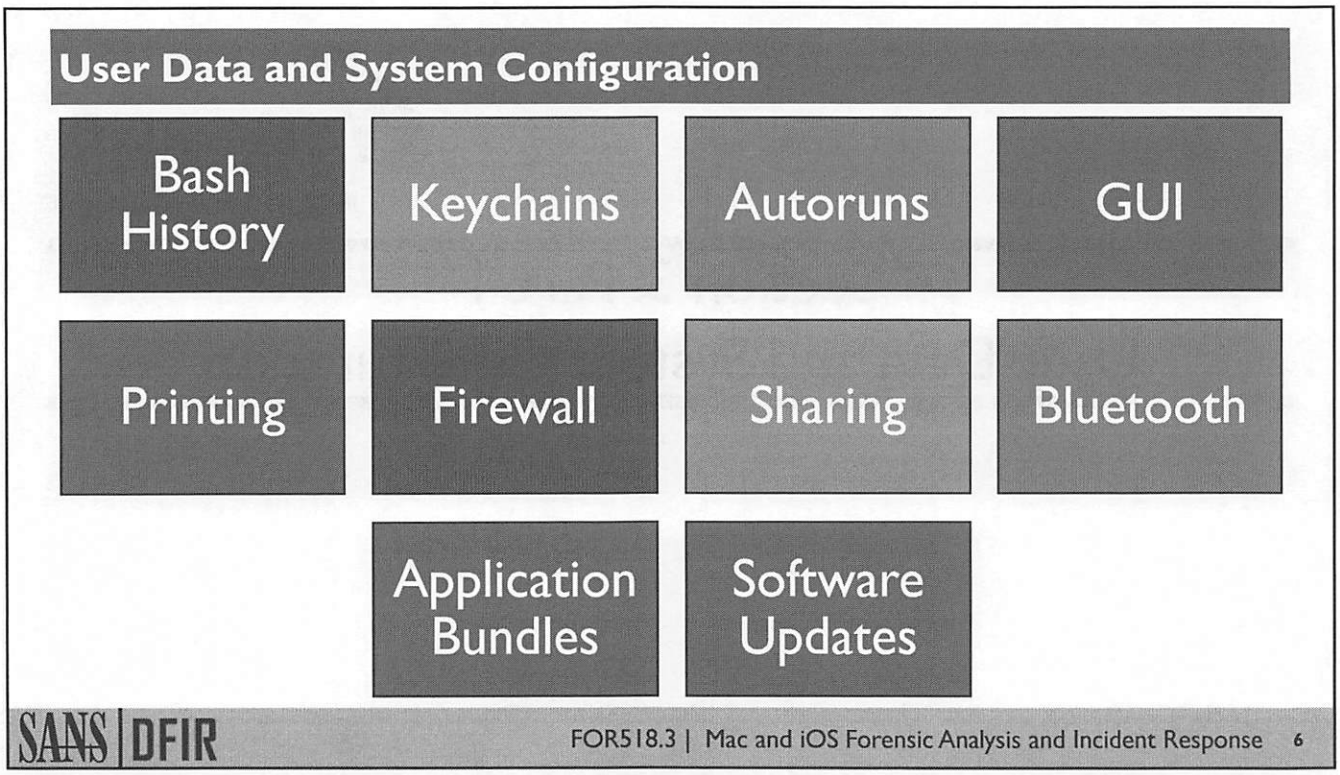
This page intentionally left blank.



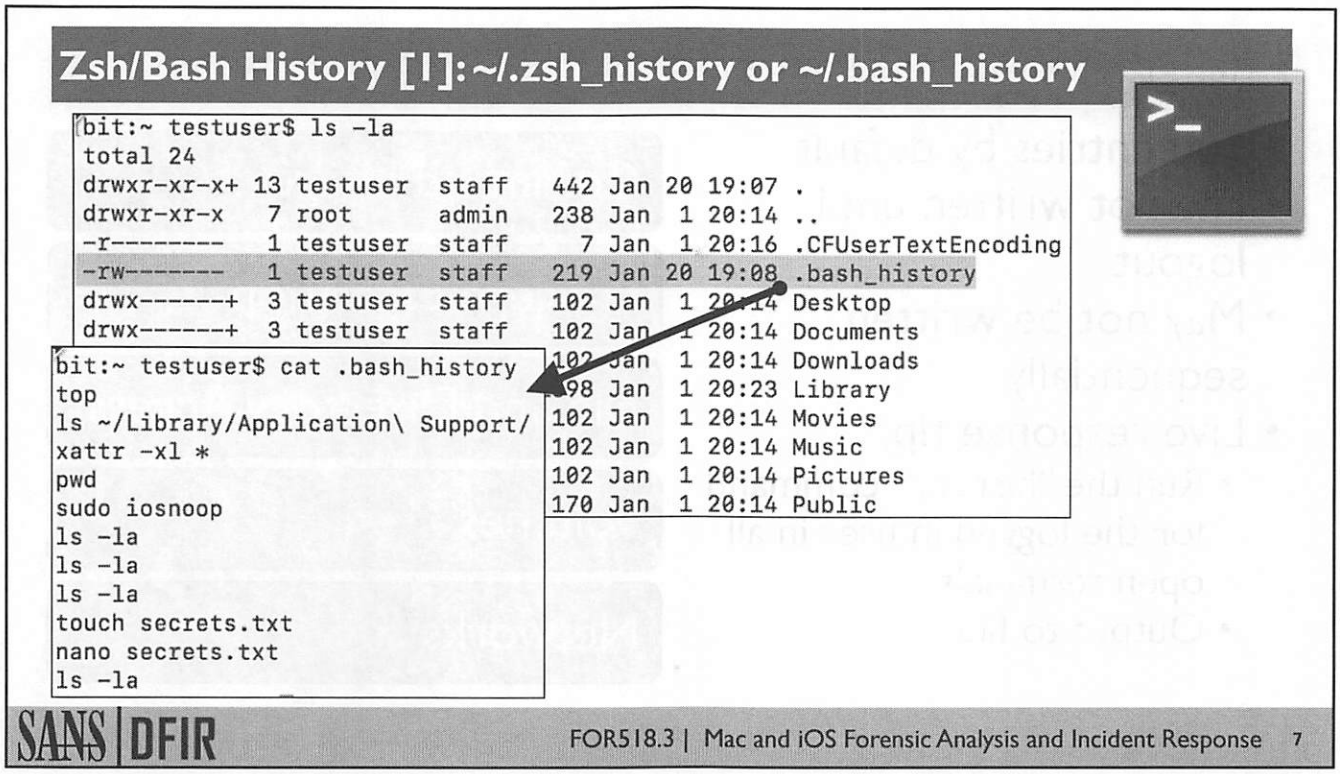
This page intentionally left blank.



This page intentionally left blank.



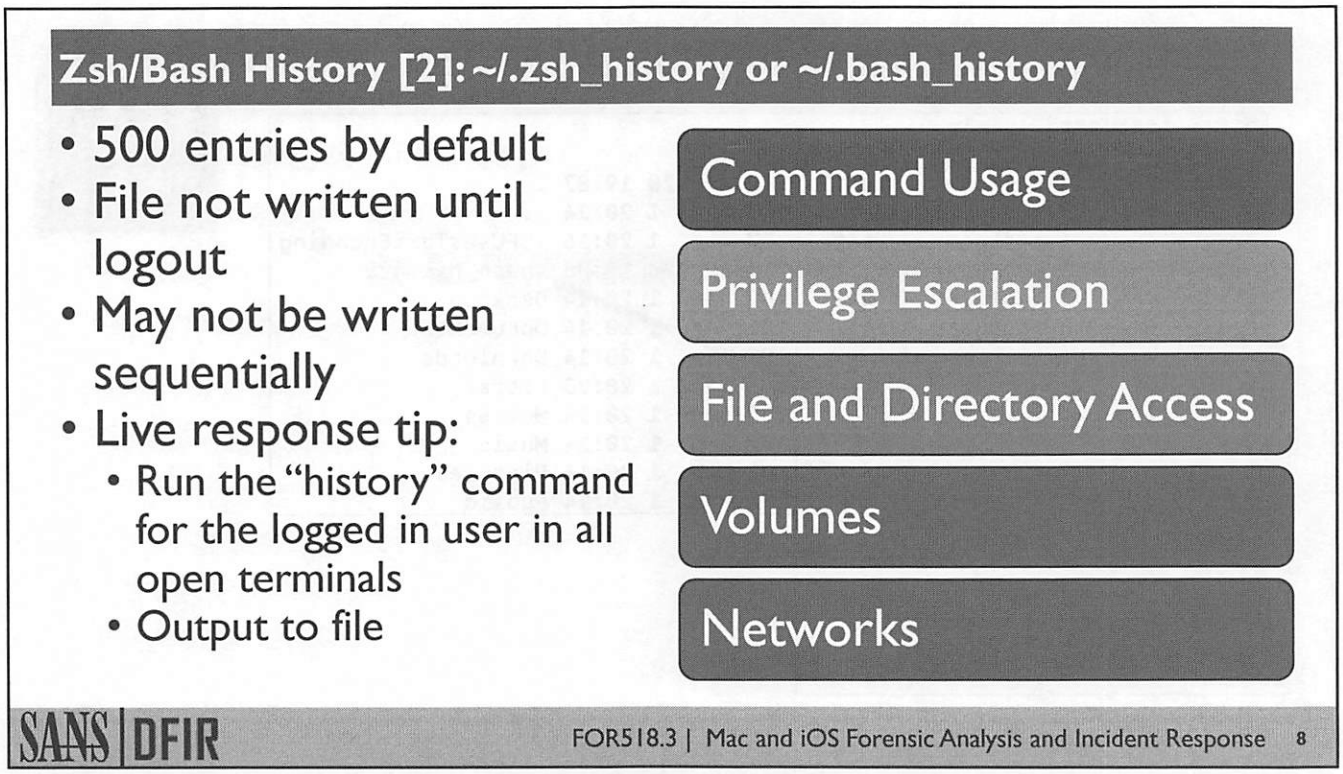
This page intentionally left blank.



The bash/zsh command history is located in the respective home directory for each user as a hidden file.

- `/Users/<user>/.bash_history`
- `/Users/<user>/.zsh_history (10.15+)`

This file is a plaintext file containing commands that were run in order of execution; however, not necessarily all in temporal order. This file will be created when the Terminal application has been used for the first time. If a user has never used the Terminal application, this file will not be created.



This file contains the command history from the Bash/Zsh shell. By default, the history file contains a maximum of 500 commands.

These commands can be useful in showing:

- What types of applications or programs the user was accessing;
- If privileges were escalated to perform administrative functions or possible suspicious actions;
- Files and directories accessed;
- Mounted volumes;
- Systems and networks that were accessed;
- Etc.

Each command is shown respective to other commands that were previously entered. It should be noted that each command has no date or time context associated in this file.

From an incident response perspective, the .zsh_history/`.bash_history` file is not updated until the user is logged out.

The history can be viewed on a live system by using the `history` command; this will show the history of the current user.

Bash Sessions [10.11-10.14]: ~/.bash_sessions/<GUID>.history

```
word:.bash_sessions oompa$ pwd
/Users/oompa/.bash_sessions
word:.bash_sessions oompa$ ls -lt
total 1784
-rw-r--r--  1 oompa  staff      0 Mar  8 16:10 3B801600-2F62-4BB7-BFB1-0C0778E98D97.historynew
-rw-r--r--  1 oompa  staff  16287 Mar  7 19:15 106C1813-7FE7-4530-BBB1-B8F5EE222FD0.history
-rw-r--r--  1 oompa  staff      0 Mar  7 19:15 106C1813-7FE7-4530-BBB1-B8F5EE222FD0.historynew
-rw-r--r--  1 oompa  staff     51 Mar  7 19:15 106C1813-7FE7-4530-BBB1-B8F5EE222FD0.session
-rw-r--r--  1 oompa  staff  16809 Mar  7 19:15 E49C5372-5D56-4A20-9A6E-FB2940FA7E08.history
-rw-r--r--  1 oompa  staff      0 Mar  7 19:15 E49C5372-5D56-4A20-9A6E-FB2940FA7E08.historynew
-rw-r--r--  1 oompa  staff     51 Mar  7 19:15 E49C5372-5D56-4A20-9A6E-FB2940FA7E08.session
-rw-r--r--  1 oompa  staff  17057 Mar  7 19:15 41FA962F-8614-4128-AF1F-1A121C775308.history
-rw-r--r--  1 oompa  staff      0 Mar  7 19:15 41FA962F-8614-4128-AF1F-1A121C775308.historynew
```

```
word:~ oompa$ exit
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.
Deleting expired sessions...27 completed.

[Process completed]
```

```
19:15 41FA962F-8614-4128-AF1F-1A121C775308.session
21:50 1111DC35-B5FE-4370-AAFE-781E4EB8D1F9.historynew
18:32 1DCE5210-2ADA-48FB-A310-541AC5AC7183.history
18:32 1DCE5210-2ADA-48FB-A310-541AC5AC7183.historynew
18:32 1DCE5210-2ADA-48FB-A310-541AC5AC7183.session
18:32 A1F3337D-7BCB-4228-9699-5F780D94D2B8.historynew
11:14 DB15993F-7755-4325-9CFB-0E943C740AFB.history
11:14 DB15993F-7755-4325-9CFB-0E943C740AFB.historynew
11:14 DB15993F-7755-4325-9CFB-0E943C740AFB.session
11:14 5DBC4C1C-AC75-42CE-A490-F1F66EAFE820.history
11:14 5DBC4C1C-AC75-42CE-A490-F1F66EAFE820.historynew
```

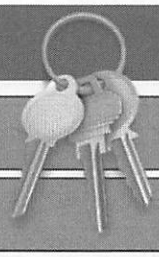

Introduced in 10.11 (and now gone in 10.15 with the introduction of zsh!), we now have session-based data (which includes file system timestamps!) for Terminal sessions.

Each `<GUID>.history` file contains the plaintext commands the user performed while in that terminal session. This does not appear to save session history for all time and is at some point removed; these files appear to get removed after about three weeks.

The `*.history` files appear to only get written once the terminal is closed/exited ("`login`" process exited).

Reference:
`/etc/bashrc_Apple_Terminal`

User and System Keychains



Store sensitive data

macOS

- User: ~/Library/Keychains
  - login.keychain-db [10.12 added "-db" extension]
  - iCloud: <Hardware UUID>/keychain-2.db
- System: /Library/Keychains
  - System.keychain

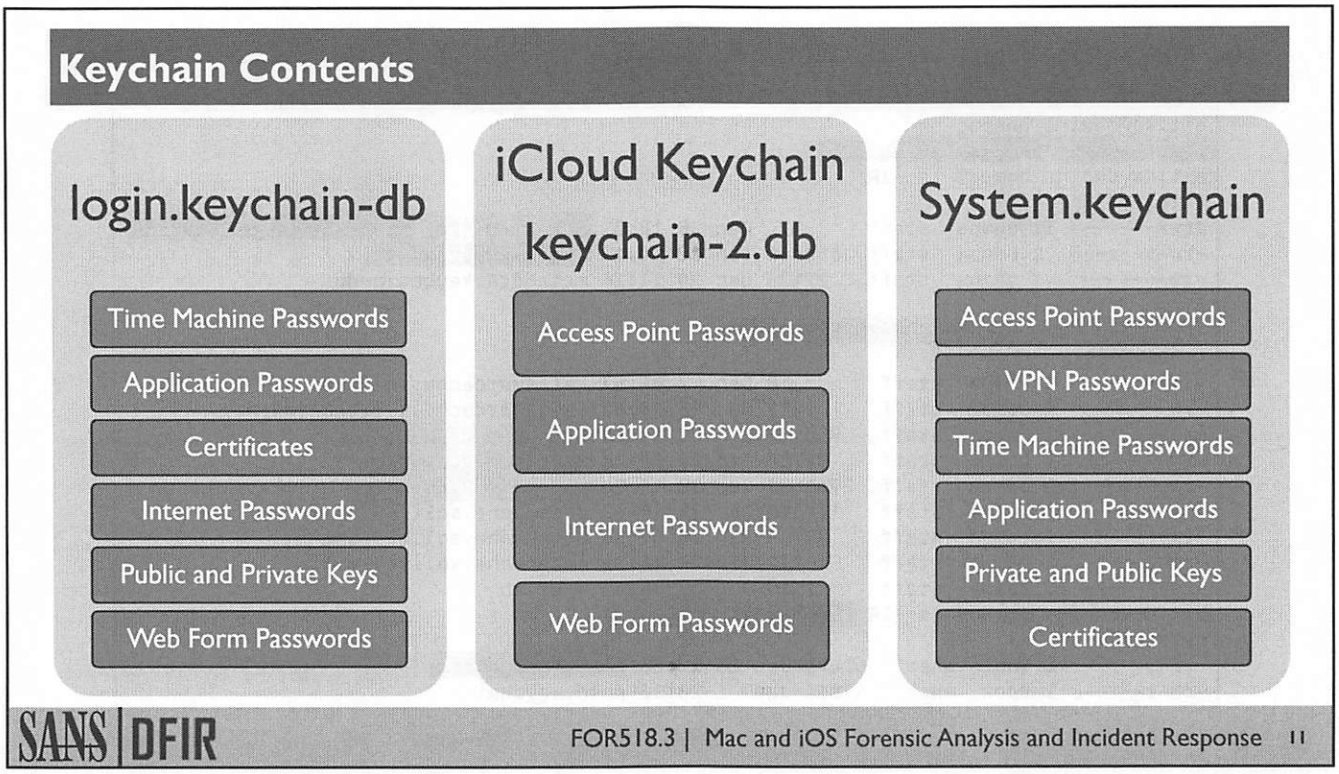
iOS (+/- iCloud Keychain)

- iOS backups
  - /Keychains: keychain-backup.plist
- Physical
  - /private/var/Keychains/keychain-2.db

The keychains on a system are used to store sensitive data such as usernames, passwords, and public and private keys. The `login.keychain[-db]` holds user-sensitive items, and the `System.keychain` holds sensitive system-wide information. Other keychains may exist on the system, including those that are user created.

Another type of keychain to be aware of is the iCloud keychain. If the user chooses to, they can sync their credentials across all of their devices, including laptops, desktops, and iDevices. If enabled, the iCloud keychain will be stored in another GUID-named (Hardware UUID) directory in the `keychain-2.db` SQLite database file.

On iOS devices, the keychains are stored in `/private/var/Keychains`. The user and/or iCloud keychain will be named `keychain-2.db` and is a SQLite database. On iOS backups this database is a plist file named `keychain-backup.plist`.



The two most important keychains are the user's `login.keychain-db` and the `System.keychain`.

The `login.keychain` may contain the user's passwords for access points, Time Machine, applications, and websites. It also stores the user's certificates and public and private keys. Among all of this information, there may also be notes if the user chooses to create them here. The default `login.keychain` password is the user's account password.

The iCloud keychain stores information that may not only be used on OS X but on iDevices as well—an access point used on the suspect's iPhone may be synced to the iCloud keychain.

The `System.keychain` also contains passwords for VPNs, access points, Time Machine, and applications. It may also contain private and public keys and certificates.

The `metadata.keychain`, if you are curious, contains data related to the user's Spotlight metadata index.

Keychain Disk Layout: macOS

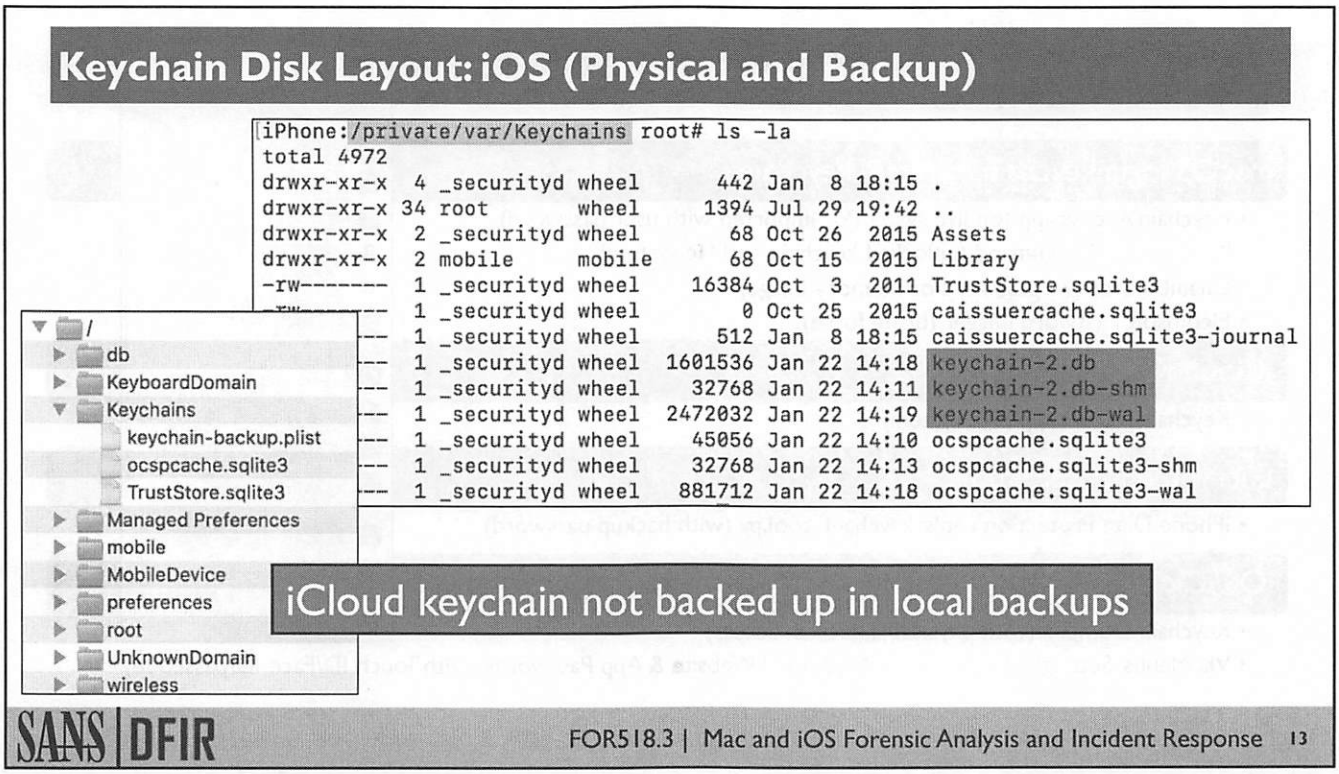
```
[bit:Keychains oompa$ pwd
 /Users/oompa/Library/Keychains
[bit:Keychains oompa$ ls -lR
 total 1328
 drwx------  11 oompa  staff     374 Dec 30 10:01 502A32AD-3CAF-5585-BC09-053E285901C8
 -rw-r--r--@  1 oompa  staff  650440 Jan 22 14:14 login.keychain-db
 -rw-------   1 oompa  staff   27176 Dec 26 11:16 metadata.keychain-db

 ./502A32AD-3CAF-5585-BC09-053E285901C8:
 total 8952
 -rw-------  1 oompa  staff        0 Dec  7 09:57 caissuercache.sqlite3
 -rw-------  1 oompa  staff      512 Dec 30 10:02 caissuercache.sqlite3-journal
 -rw-------  1 oompa  staff   913408 Jan 22 14:25 keychain-2.db
 -rw-------  1 oompa  staff    32768 Jan 20 09:00 keychain-2.db-shm
 -rw-------  1 oompa  staff  2966432 Jan 22 14:26 keychain-2.db-wal
 -rw-------  1 oompa  staff   114688 Jan 21 10:43 ocspcache.sqlite3
 -rw-------  1 oompa  staff    32768 Jan 20 09:00 ocspcache.sqlite3-shm
 -rw-------  1 oompa  staff   510912 Jan 22 14:34 ocspcache.sqlite3-wal
 -rw-------  1 oompa  staff     1518 Dec 10 18:41 user.kb
[bit:Keychains oompa$ ls -lR /Library/Keychains/
 total 232
 -rw-r--r--  1 root  wheel  61476 Jan 13 12:26 System.keychain
 -rw-r--r--@ 1 root  wheel  49864 Jan  1 20:16 apsd.keychain
```


On macOS, the keychains are stored in the user's Library directory (`~/Library/Keychains/`) as `login.keychain[-db]` or `keychain-2.db`.

10.12 added a new file extension to the login.keychain file, "`-db`". Other keychain-related security items also exist in these directories.

The iCloud keychain, if the user chooses to use it, will be stored under a GUID (the Hardware UUID of the system), in the `keychain-2.db` file (same file as on iOS physical devices). While the user may not be actively using the keychain, it still may exist on their system.

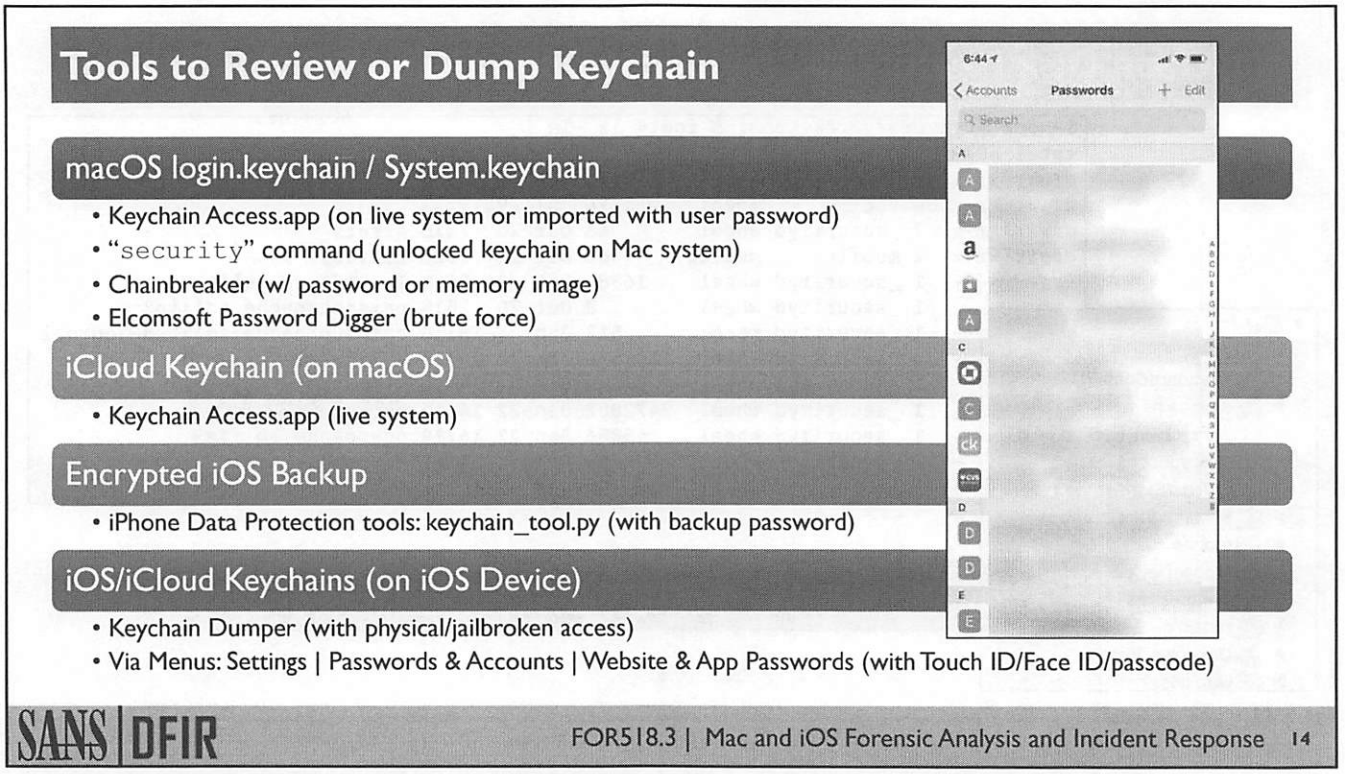
The `System.keychain` file is stored in the System Library (`/Library/Keychains/` directory).



The screenshots above show the layout for the keychains on iOS; the `keychain-2.db` will only be available on physical iOS device acquisitions—otherwise you will see a `keychain-backup.plist` file in backup acquisitions.

On iOS disks, the keychain files are located in `/private/var/Keychains`, and, just like macOS, there may be a few other keychain-related items in the directory; however, `keychain-2.db` contains most of the user-based sensitive items.

On iOS backups, the keychain-backup.plist file may be stored in a directory named "`Keychains`" or "`KeychainDomain`", depending on the acquisition tool used. It is worth noting that iCloud keychain items are not specifically backed up in this plist file.



Different tools can be used to dump or access the keychain for review.

On macOS systems, you can use Keychain Access.app or the native "`security`" command to view the passwords. A password may or may not be needed, depending on if you are viewing on the live system (say during system triage) or reviewing a dead drive keychain file on a separate analyst system. Other tools like Chainbreaker or Elcomsoft Password Digger can be used offline with/without the password.

iCloud keychains are the toughest to get into right now, as there are limited tools to do so. This keychain can be reviewed using the Keychain Access.app on a live system; however, it can't be used to import onto another system.

Encrypted iPhone Backup keychains can be extracted by using the iPhone Data Protection tools, assuming you have the iPhone backup password.
Follow steps 1–3 from `https://dpron.com/recovering-google-authenticator-keys-from-ios-backups/`. Additional help from `http://apple.stackexchange.com/questions/190139/view-keychain-from-encrypted-ios-8-2-backup` may be needed.

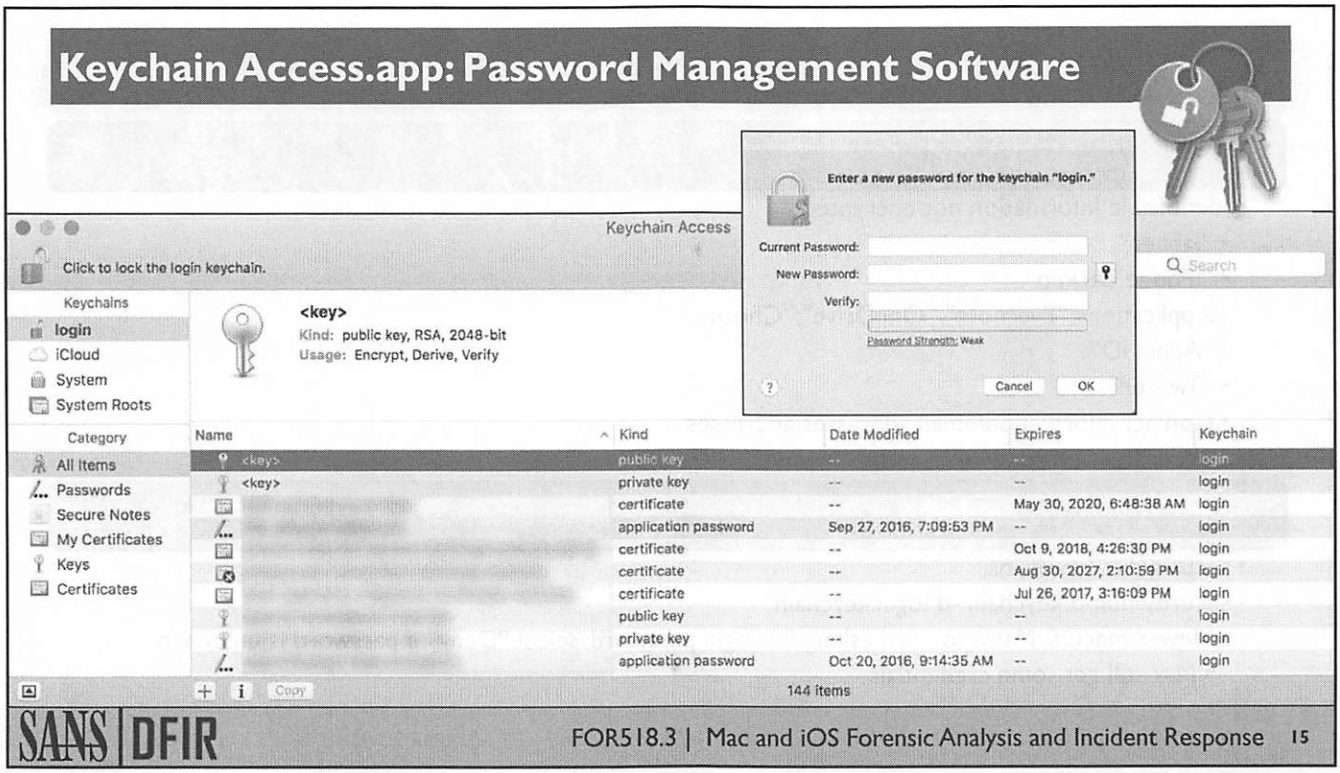
Once set up, you can run the command below to dump the contents of the keychain.

```
python keychain_tool.py -d keychain-backup.plist Manifest.plist
```

Other commercial tools like Elcomsoft Phone Breaker—Encrypted (with Backup Password) or Unencrypted (with "Security Key" extracted from the physical device)—can be used to dump the iOS keychain.

To dump the SQLite version of the iOS/iCloud keychain, you can use an iOS device and the Keychain Dumper script.

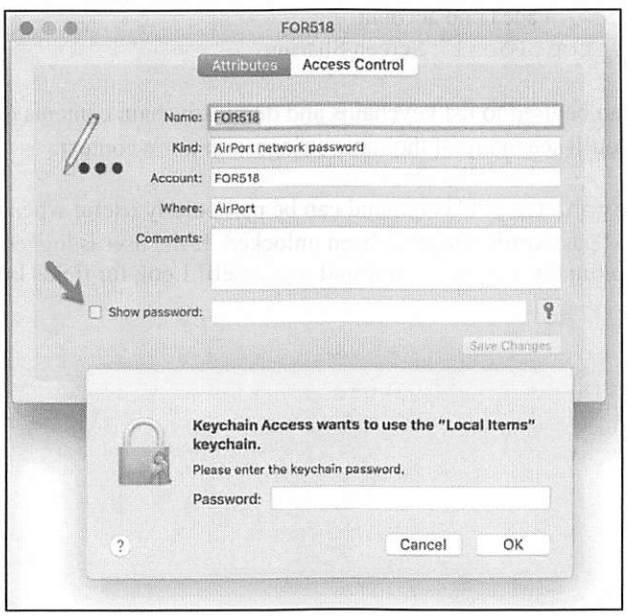
References:
iPhone Data Protection Tools: `https://github.com/dinosec/iphone-dataprotection`
Chainbreaker: `https://github.com/n0fate/chainbreaker`
Elcomsoft Password Digger: `https://www.elcomsoft.com/epd.html`
Keychain Dumper: `https://github.com/ptoomey3/Keychain-Dumper`



The `Keychain Access` application can be used to interact with the keychains. This application works like any other password management system. Click on the keychain in the upper right, and the contents will be shown on the main screen. Be sure to choose "`All Items`" in the `Category` listing; otherwise, you may not see all the data stored.

Note: The default password for a user's `login.keychain` is their login password; however, this can be changed to anything in the `Keychain Access` application menu, as shown in the inset screenshot.

Below is an example of when a password is selected for viewing. The analyst can select the "`Show password`" checkbox and input the user's keychain password (by default, the login password).



Other Keychain Access

"strings *" keychain

- Identifiable Information not encrypted
  - "Jabber"
  - "iPhone Backup"
  - Applications: "Evernote", "OneDrive", "Chrome"
  - "Apple ID"
  - "Twitter"
  - Contact information/email addresses/addresses
- Certificates and passwords are encrypted

"security" command

- security list-keychains
- security dump-keychain -d login.keychain
  - Newer macOS: Many pop-ups asking for login password, select "Deny" if password is unknown.
    - May still get some credentials

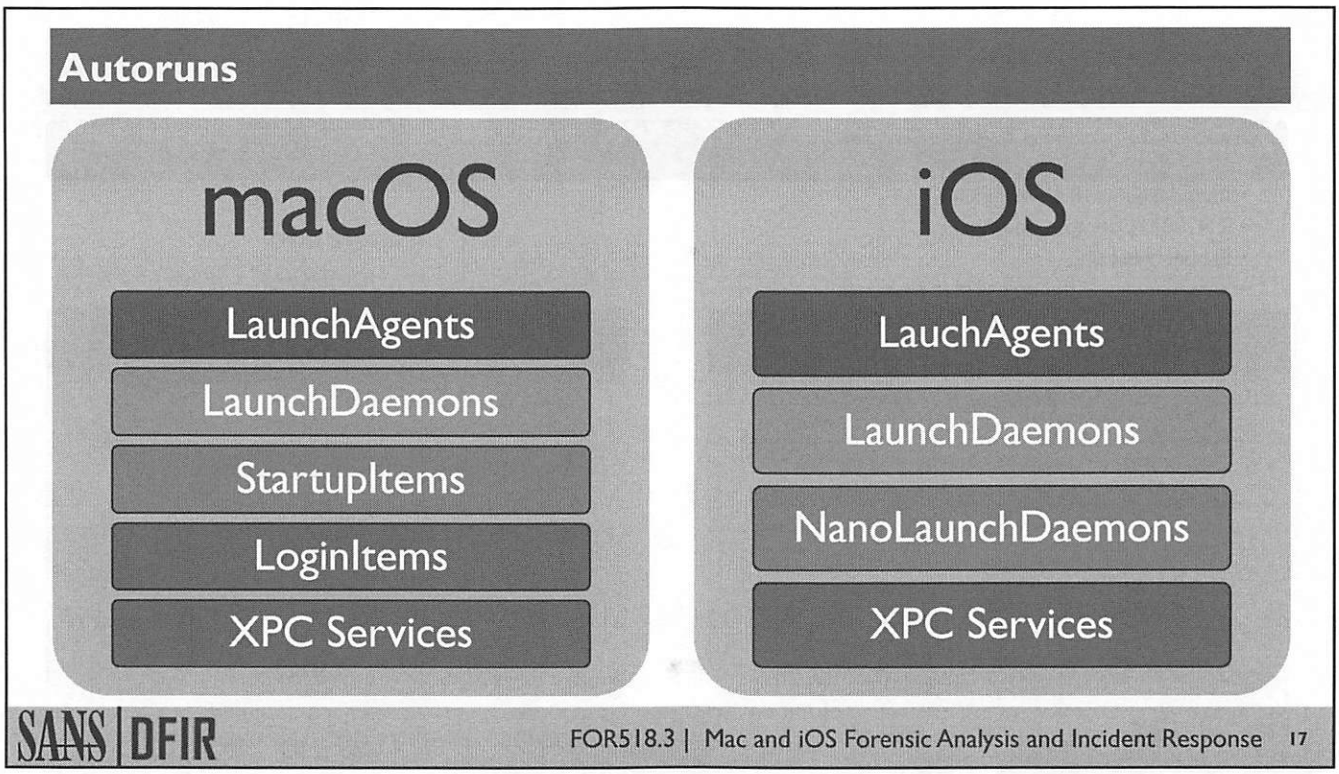
The `strings` command can be used to identify interesting keywords in a keychain that are not encrypted. This can give you an idea of what might be stored in the keychain if you are unable to access it with the user's password (remember a user does not have to use their login password, but it is used by default).

Strings like the following may suggest certain items are saved in the keychain:

- "`imagent`": iChat
- "`iCloud`": iCloud account
- "`com.apple.account.google`": Google account
- "`Jabber: user@gmail.com`": Google Chat account
- "`AIM: <userhandle>`": AIM account
- "`FaceTime: <acct>`": FaceTime account
- "`<system>._rfb._tcp.local`": Screen Sharing

The `security` command can be used to list keychains and dump keychain contents (though not the encrypted contents) on a live system. You'll see many of the same strings with more context.

The "`security dump-keychain -d`" command can be particularly useful when used on a live system—it can sometimes dump plaintext passwords that have been unlocked. If the user is logged in but will not give up their password, you may be able to run this command and find it yourself! Look for fields labeled "`data:`".



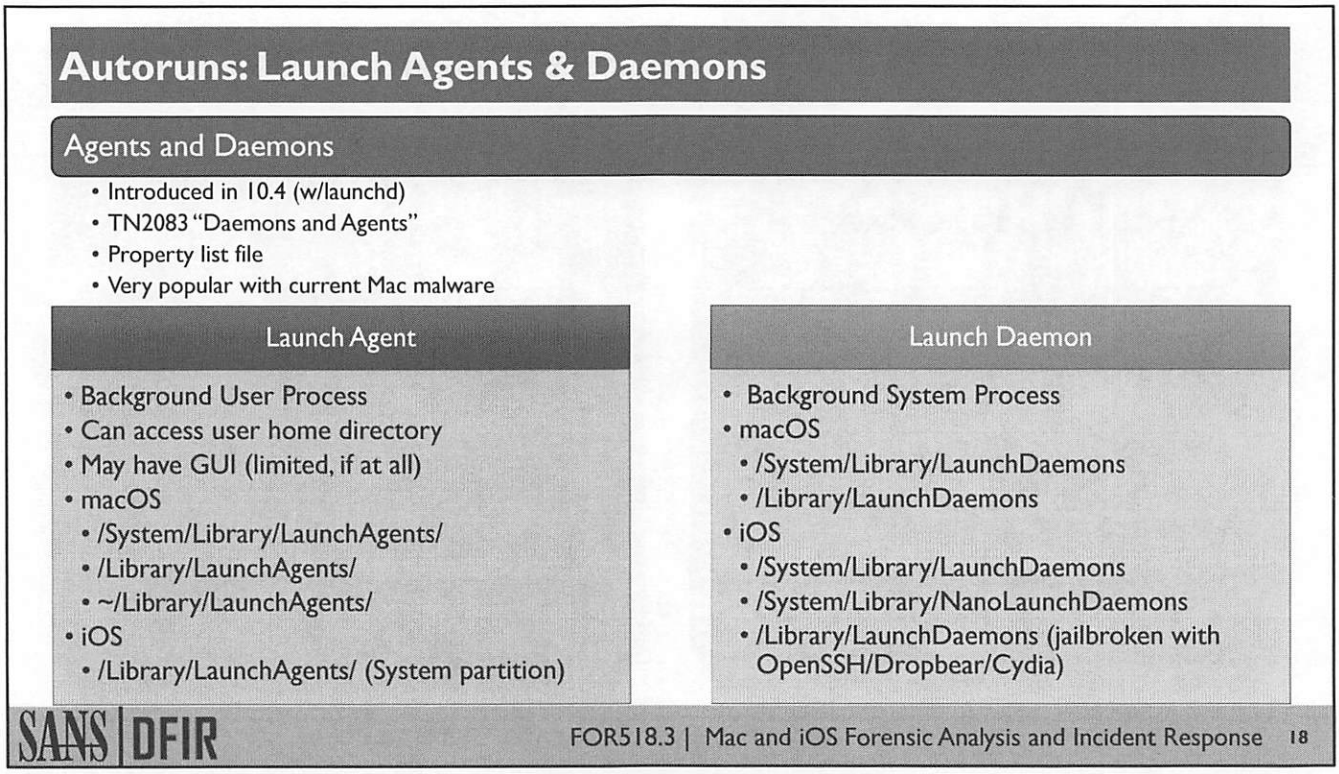
macOS and iOS systems have a variety of locations that applications can auto start from; however, not nearly as many as Windows systems.

While these locations can be used legitimately, malware authors also tend to use them—especially on macOS.

Some tools you might want to look at to find autoruns and auditing information:

- Knockknock: `github.com/synack/knockknock`
- OSXAuditor: `github.com/jipegit/OSXAuditor`
- checkout4mac: `code.google.com/p/checkout4mac/`
- pac4mac: `code.google.com/p/pac4mac/`
- osxautoruns: `http://www.malicious-streams.com/Downloads/files/6bad73d5437c84b23fde9ebad97c7cff-6.html`

Reference:
Joel Yonts, *Mac OS X Malware Analysis*, Malicious Streams
`http://www.malicious-streams.com//article/Mac_OSX_Malware_Analysis.pdf`



The preferred method of creating persistence is by using Launch Agents and Daemons. These methods were introduced in Mac OS X 10.4 with the introduction of `launchd`, the system agent and daemon launch manager. The agents and daemons are implemented as information in a property list file. Most modern Mac malware use these methods to keep their malicious files persistent across system shutdowns and reboots.

Launch Agents are a background user process, while daemons are background system processes. These processes can access user home directories and have limited GUI interfaces, while daemons cannot. The typical location where `Launch Agents` are found is in the `LaunchAgents` directory, located in the System Library, Local Library, and User Library directories. Those agents launched from `the /System/Library` or `/Library/` directories are launched for all users, while those located in the Users `Library` directory are available only to that user. iOS devices may have them located in `/Library/LaunchAgents` on the System partition.

Launch Daemons are a background system process, while agents are background user processes. The typical location where Launch Daemons are found is in the `LaunchDaemons` directory, located in the System Library and Local Library directories. On iOS, Launch Daemons may provide insight into what processes are running—these areas are particularly interesting on jailbroken devices where various programs are launched from here, such as SSH servers (OpenSSH/Dropbear) and app stores like Cydia.

Reference:

Daemons and Agents, Tech Note 2083 (TN2083)

`http://developer.apple.com/library/mac/#technotes/tn2083/_index.html`

Autoruns: Launch Agent Examples [1]

com.apple.AOSNotificationOSX.plist	com.apple.SystemUIServer.plist
com.apple.AddressBook.SourceSync.plist	com.apple.TMLaunchAgent.plist
com.apple.AddressBook.abd.plist	com.apple.TrustEvaluationAgent.plist
com.apple.AirPortBaseStationAgent.plist	com.apple.UserEventAgent-Aqua.plist
com.apple.AppStoreUpdateAgent.plist	com.apple.UserEventAgent-LoginWindow.plist
com.apple.AppleGraphicsWarning.plist	com.apple.UserNotificationCenterAgent-LoginWindow.plist
com.apple.BezelUI.plist	com.apple.UserNotificationCenterAgent.plist
com.apple.CoreLocationAgent.plist	com.apple.VoiceOver.plist
com.apple.DictionaryPanelHelper.plist	com.apple.WebKit.PluginAgent.plist
com.apple.DiskArbitrationAgent.plist	com.apple.ZoomWindow.plist
com.apple.Dock.plist	com.apple.alf.useragent.plist
com.apple.FTCleanup.plist	com.apple.aos.migrate.plist
com.apple.FileSyncAgent.PHD.plist	com.apple.bluetoothUIServer.plist
com.apple.FileSyncAgent.iDisk.plist	com.apple.btsa.plist
com.apple.Finder.plist	com.apple.cfnetwork.AuthBrokerAgent.plist
com.apple.FontRegistryUIAgent.plist	com.apple.cookied.plist
com.apple.FontValidator.plist	com.apple.coredata.externalrecordswriter.plist
com.apple.FontValidatorConduit.plist	com.apple.coreservices.appleid.authentication.plist
com.apple.FontWorker.plist	com.apple.coreservices.uiagent.plist
com.apple.KerberosHelper.LKDCHelper.plist	com.apple.csuseragent.plist
com.apple.LaunchServices.lsboxd.plist	com.apple.cvmsCompAgent_i386.plist
com.apple.NetworkDiagnostics.plist	com.apple.cvmsCompAgent_x86_64.plist
com.apple.PCIESlotCheck.plist	com.apple.distnoted.xpc.agent.plist
com.apple.PreferenceSyncAgent.plist	com.apple.familycontrols.useragent.plist
com.apple.PubSub.Agent.plist	com.apple.findmymacmessenger.plist
com.apple.ReclaimSpaceAgent.plist	com.apple.fontd.useragent.plist
com.apple.RemoteDesktop.plist	com.apple.gssd-agent.plist
com.apple.ReportCrash.Self.plist	com.apple.helpd.plist
com.apple.ReportCrash.plist	com.apple.iCalPush.plist
com.apple.ReportGPURestart.plist	com.apple.iChat.Theater.plist
com.apple.ReportPanic.plist	com.apple.imagent.plist
com.apple.ScreenReaderUIServer.plist	com.apple.imklaunchagent.plist
com.apple.ServiceManagement.LoginItems.plist	com.apple.imtranscoderagent.plist
com.apple.SubmitDiagInfo.plist	com.apple.imtransferagent.plist

The screenshot above shows a typical example of the `LaunchAgents` directory in `/System/Library/`.

Each launch agent property list is named in the reverse DNS format.

com.apple.AOSNotificationOSX.plist	com.apple.SystemUIServer.plist
com.apple.AddressBook.SourceSync.plist	com.apple.TMLaunchAgent.plist
com.apple.AddressBook.abd.plist	com.apple.TrustEvaluationAgent.plist
com.apple.AirPortBaseStationAgent.plist	com.apple.UserEventAgent-Aqua.plist
com.apple.AppStoreUpdateAgent.plist	com.apple.UserEventAgent-LoginWindow.plist
com.apple.AppleGraphicsWarning.plist	com.apple.UserNotificationCenterAgent-LoginWindow.plist
com.apple.BezelUI.plist	com.apple.UserNotificationCenterAgent.plist
com.apple.CoreLocationAgent.plist	com.apple.VoiceOver.plist
com.apple.DictionaryPanelHelper.plist	com.apple.WebKit.PluginAgent.plist
com.apple.DiskArbitrationAgent.plist	com.apple.ZoomWindow.plist
com.apple.Dock.plist	com.apple.alf.useragent.plist
com.apple.FTCleanup.plist	com.apple.aos.migrate.plist
com.apple.FileSyncAgent.PHD.plist	com.apple.bluetoothUIServer.plist
com.apple.FileSyncAgent.iDisk.plist	com.apple.btsa.plist
com.apple.Finder.plist	com.apple.cfnetwork.AuthBrokerAgent.plist
com.apple.FontRegistryUIAgent.plist	com.apple.cookied.plist
com.apple.FontValidator.plist	com.apple.coredata.externalrecordswriter.plist
com.apple.FontValidatorConduit.plist	com.apple.coreservices.appleid.authentication.plist
com.apple.FontWorker.plist	com.apple.coreservices.uiagent.plist
com.apple.KerberosHelper.LKDCHelper.plist	com.apple.csuseragent.plist
com.apple.LaunchServices.lsboxd.plist	com.apple.cvmsCompAgent_i386.plist
com.apple.NetworkDiagnostics.plist	com.apple.cvmsCompAgent_x86_64.plist
com.apple.PCIESlotCheck.plist	com.apple.distnoted.xpc.agent.plist
com.apple.PreferenceSyncAgent.plist	com.apple.familycontrols.useragent.plist
com.apple.PubSub.Agent.plist	com.apple.findmymacmessenger.plist
com.apple.ReclaimSpaceAgent.plist	com.apple.fontd.useragent.plist
com.apple.RemoteDesktop.plist	com.apple.gssd-agent.plist
com.apple.ReportCrash.Self.plist	com.apple.helpd.plist
com.apple.ReportCrash.plist	com.apple.iCalPush.plist
com.apple.ReportGPURestart.plist	com.apple.iChat.Theater.plist
com.apple.ReportPanic.plist	com.apple.imagent.plist
com.apple.ScreenReaderUIServer.plist	com.apple.imklaunchagent.plist
com.apple.ServiceManagement.LoginItems.plist	com.apple.imtranscoderagent.plist
com.apple.SubmitDiagInfo.plist	com.apple.imtransferagent.plist

Autoruns: Launch Agent Examples [2]

▼ Root	Dictionary	(4 items)
Label	String	com.openssh.ssh-agent
▼ ProgramArguments	Array	(2 items)
Item 0	String	/usr/bin/ssh-agent
Item 1	String	-l
▼ Sockets	Dictionary	(1 item)
▼ Listeners	Dictionary	(1 item)
SecureSocketWithKey	String	SSH_AUTH_SOCK
EnableTransactions	Boolean	YES

▼ Root	Dictionary	(6 items)
▼ MachServices	Dictionary	(1 item)
com.bjango.istatmenusagent	Boolean	YES
LimitLoadToSessionType	String	Aqua
Label	String	com.bjango.istatmenus.agent
▼ KeepAlive	Dictionary	(2 items)
Crashed	Boolean	YES
SuccessfulExit	Boolean	NO
Program	String	/Library/Application Support/iStat Menus 6/iStatMenusAgent.app/Contents/MacOS/iStatMenusAgent

Examples of launch agents are shown above. Each launch agent contains two basic keys:

- `Program` or `ProgramArguments`: File path of the program to launch. This key can have multiple items; the first is always the program to execute, while the others are program arguments. This example runs the command `/usr/bin/ssh-agent` with the `-l` flag.
- `Label`: Reverse DNS name for the process. This string should be unique across launch agents.

Other keys may be used to customize the launch agent. Many of these keys are explained in the man page for `launchd.plist`.

The `KeepAlive` key contains the subkey "`SucessfulExit`". According to Tech Note 2083, this means to "run on demand as long as you exit successfully".

Reference:
`launchd.plist` man Page

Autoruns: Launch Daemon Example

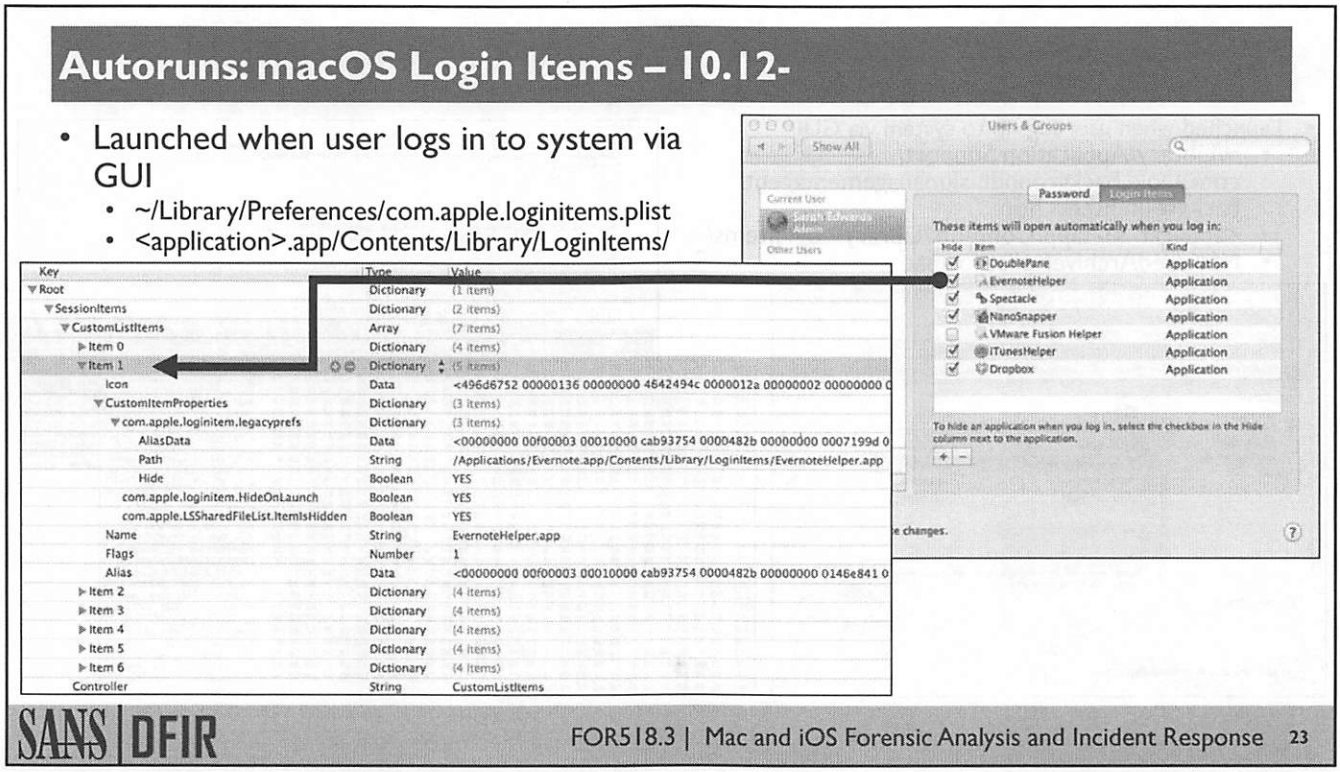
```
Sat Dec  9 09:46:22 EST 2017

Removing old temporary files:

Cleaning out old system announcements:

Removing stale files from /var/rwho:

Disk status:
Filesystem     Size   Used  Avail Capacity iused               ifree %iused  Mounted on
/dev/disk1s1  932Gi  559Gi  369Gi    61% 2170221 9223372036852685586    0%   /
/dev/disk1s4  932Gi  3.0Gi  369Gi     1%       5 9223372036854775802    0%   /private/var/vm
/dev/disk2s2   95Mi   55Mi   40Mi    58%     922          4294966357    0%   /Volumes/iExplorer

Network interface status:
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16384 <Link#1>                         53436     0    53436     0     0
lo0   16384 127           localhost          53436     -    53436     -     -
lo0   16384 localhost   ::1                  53436     -    53436     -     -
lo0   16384 fe80::1%lo0 fe80:1::1            53436     -    53436     -     -
gif0* 1280  <Link#2>                             0     0        0     0     0
stf0* 1280  <Link#3>                             0     0        0     0     0
XHC0* 0     <Link#4>                             0     0        0     0     0
XHC20 0     <Link#5>                             0     0        0     0     0
XHC1* 0     <Link#6>                             0     0        0     0     0
en0   1500  <Link#8>    f4:0f:24:3b:a2:3b 19198042     0 49461933     0     0
en0   1500  fe80::1cde: fe80:8::1cde:86c6 19198042     - 49461933     -     -
en0   1500  10.11.12/24   sarahs-mbp.stat 19198042     - 49461933     -     -
p2p0  2304  <Link#9>    06:0f:24:3b:a2:3b        0     0        0     0     0
awdl0 1484  <Link#10>   9e:7b:72:84:53:7d       91     0       93     0     0
awdl0 1484  fe80::9c7b: fe80:a::9c7b:72ff       91     -       93     -     -
en1   1500  <Link#11>   ee:00:78:89:d1:01        0     0        0     0     0
en2   1500  <Link#12>   ee:00:78:89:d1:00        0     0        0     0     0
en3   1500  <Link#13>   ee:00:78:89:d1:05        0     0        0     0     0
en4   1500  <Link#14>   ee:00:78:89:d1:04        0     0        0     0     0
bridg 1500  <Link#15>   ee:00:78:89:d1:01        0     0        1     0     0
utun0 2000  <Link#16>                            0     0        2     0     0
utun0 2000  fe80::2010: fe80:10::2010:a1b        0     -        2     -     -
utun1 1380  <Link#17>                           74     0       58     0     0
utun1 1380  fe80::60a5: fe80:11::60a5:51b       74     -       58     -     -
utun2 1380  <Link#18>                          102     0       68     0     0
utun2 1380  fe80::db26: fe80:12::db26:de5      102     -       68     -     -
en5   1500  <Link#7>    ac:de:48:00:11:22     1634     0     1615     0     0
en5   1500  fe80::aede: fe80:7::aede:48ff     1634     -     1615     -     -

Local system status:
 9:46  up 2 days, 13:30, 1 user, load averages: 89.70 28.43 11.70

-- End of daily output --
```

▼ Root	Dictionary	(6 items)
Label	String	com.apple.periodic-daily
▼ ProgramArguments	Array	(2 items)
Item 0	String	/usr/libexec/periodic-wrapper
Item 1	String	daily
LowPriorityIO	Boolean	YES
Nice	Number	1
▼ LaunchEvents	Dictionary	(1 item)
▼ com.apple.xpc.activity	Dictionary	(1 item)
▼ com.apple.periodic-daily	Dictionary	(5 items)
Interval	Number	86,400
GracePeriod	Number	14,400
Priority	String	Maintenance
AllowBattery	Boolean	NO
Repeating	Boolean	YES
AbandonProcessGroup	Boolean	YES

Good examples of launch daemons are the Unix periodic maintenance scripts: `Daily`, `weekly`, and `monthly`.

The screenshot shows the output from the daily script in the `daily.out` file. The coordinating Launch Daemon, `com.apple.periodic-daily.plist`, contains many of the same keys as the `LaunchAgents`. The `ProgramArguments` key contains the path to the daily script with the argument "`daily`".

The property list also contains the key `LaunchEvents` keys. This shows how often to run the process; in this case, every day (86,400 seconds), with a grace period of four hours (14,400 seconds).



Each user may have a set of autoruns that get launched when a user logs in via the system GUI login. Each login item can be "hidden" on launch. This means the login item is hidden from view when it is starting up. `LoginItems`, while usually legitimate, may be used as a malware persistence mechanism.

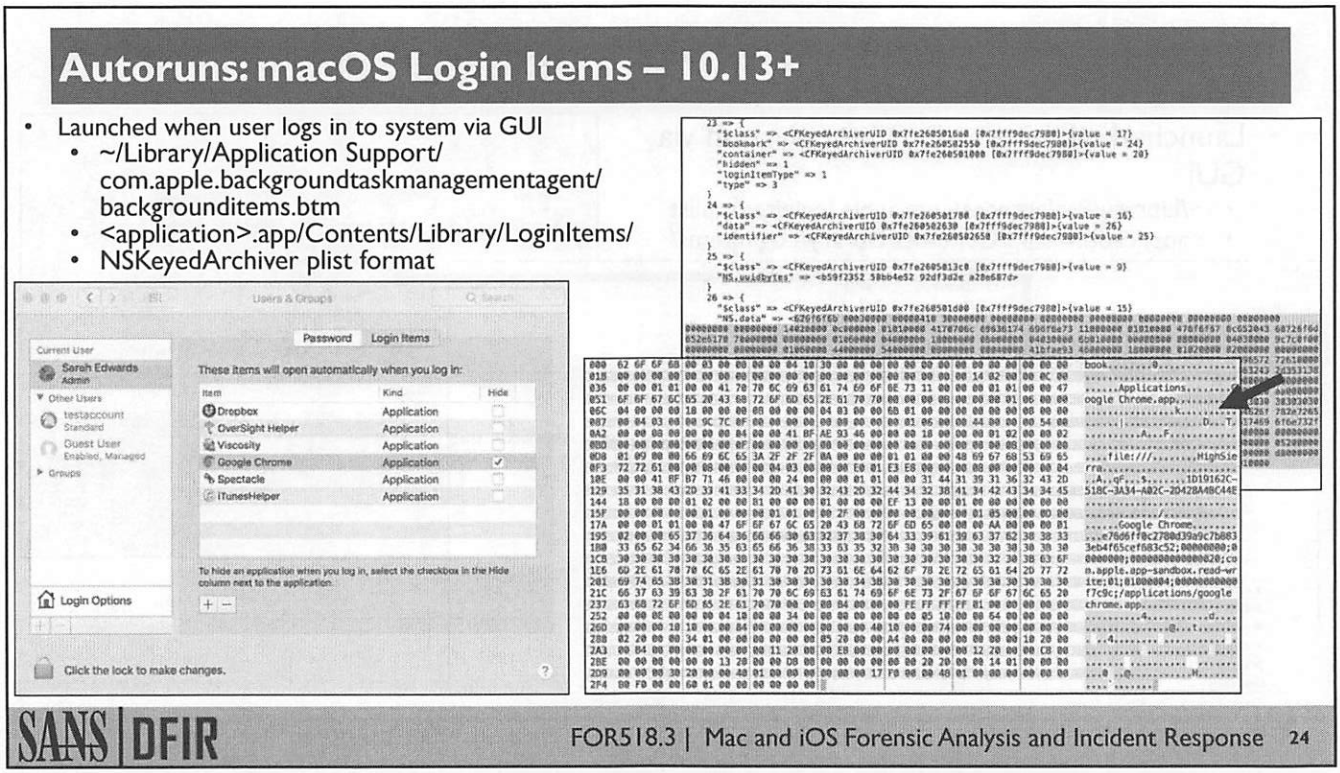
The `com.apple.loginitems.plist` contains each login item. Each item has an associated icon, alias data blob, the path to the login item, and a binary value showing if the item was "hidden" on launch.

Locations:
`~/Library/Preferences/com.apple.loginitems.plist`
`<application>.app/Contents/Library/LoginItems/`

Login items are similar to the Microsoft registry key `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`. These are usually small programs or helper applications.

Each user has their own login items listed in the `com.apple.loginitems.plist` file located in their `/Library/Preferences/` directory. The login item application is usually found within an application bundle's `Library` directory.

While rare, global login items may be listed in the `com.apple.loginwindow.plist` file in the `/Library/Preferences/` directory under the `AutoLaunchedApplicationDictionary` key.



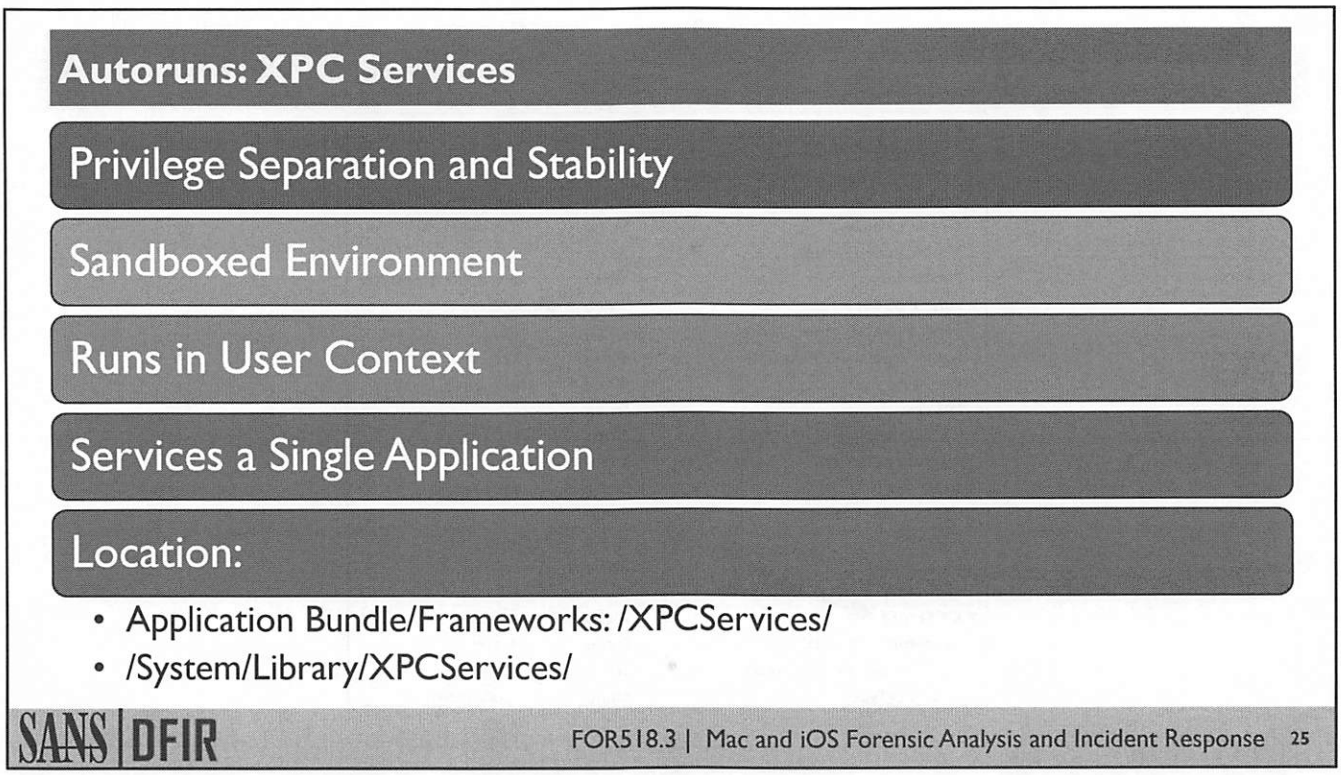
Each user may have a set of autoruns that get launched when a user logs in via the system GUI login. Each login item can be "hidden" on launch, which means the login item is hidden from view when it is starting up. `Login items`, while usually legitimate, may be used as a malware persistence mechanism.

10.13 updated the mechanism somewhat. While it is still a plist file, it is an NSKeyedArchiver plist file. It also has a new location listed below. Each user will have their own login items.

Locations:

- `~/Library/Application Support/com.apple.backgroundtaskmanagementagent/backgrounditems.btm`
- `<application>.app/Contents/Library/LoginItems/`

Login items are similar to the Microsoft registry key `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`. These are usually small programs or helper applications.



XPC (Interprocess Communication) services are used to control the stability of a system. These services will only service one application. If the service crashes, only the application will crash, rather than the whole system. These services run in the user context and are located in various `XPCServices` directories.

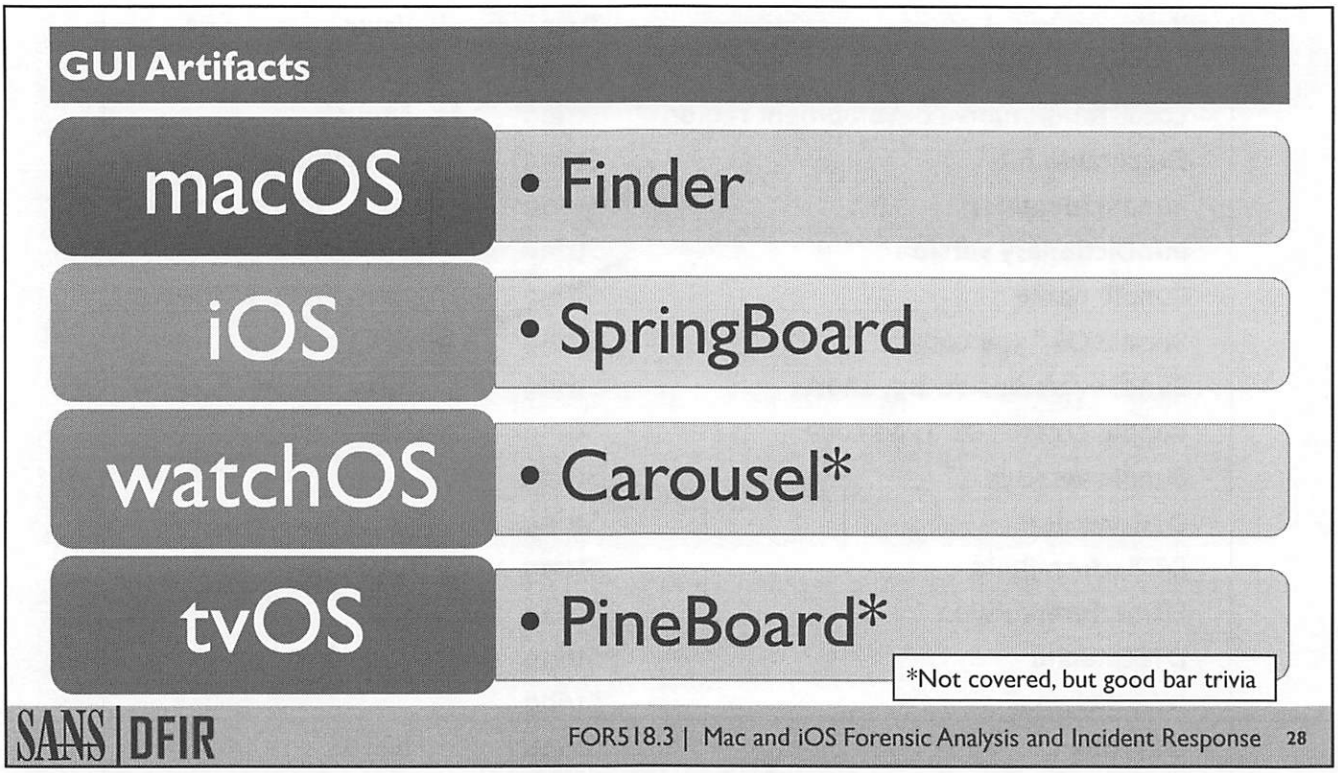
Reference:
Creating XPC Services: Apple Developer Documentation
`http://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/BPSystemStartup/Chapters/CreatingXPCServices.html`

Autoruns: XPC Service Example

Key	Type	Value
BuildMachineOSBuild	String	11D17a
Localization native development region	String	English
Executable file	String	com.apple.qtkitserver
Bundle identifier	String	com.apple.qtkitserver
InfoDictionary version	String	6.0
Bundle name	String	com.apple.qtkitserver
Bundle OS Type code	String	XPC!
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
DTCompiler	String	
DTPlatformBuild	String	11D17a
DTPlatformVersion	String	GM
DTSDKBuild	String	11D17a
DTSDKName	String	
DTXcode	String	0410
DTXcodeBuild	String	11D17a
Application is agent (UIElement)	Boolean	YES
▼XPCService	Diction...	(2 items)
▼EnvironmentVariables	Diction...	(1 item)
MallocCorruptionAbort	String	1
ServiceType	String	Application

The screenshot shows an example of an XPC service property list for the `qtkitserver` located in `/System/Library/XPCServices/com.apple.qtkitserver.xpc/Contents/Info.plist`. The property list contains the bundle type code of "`XPC!`" and a key labeled XPC service that will contain many of the same keys used in launch agents and daemons.

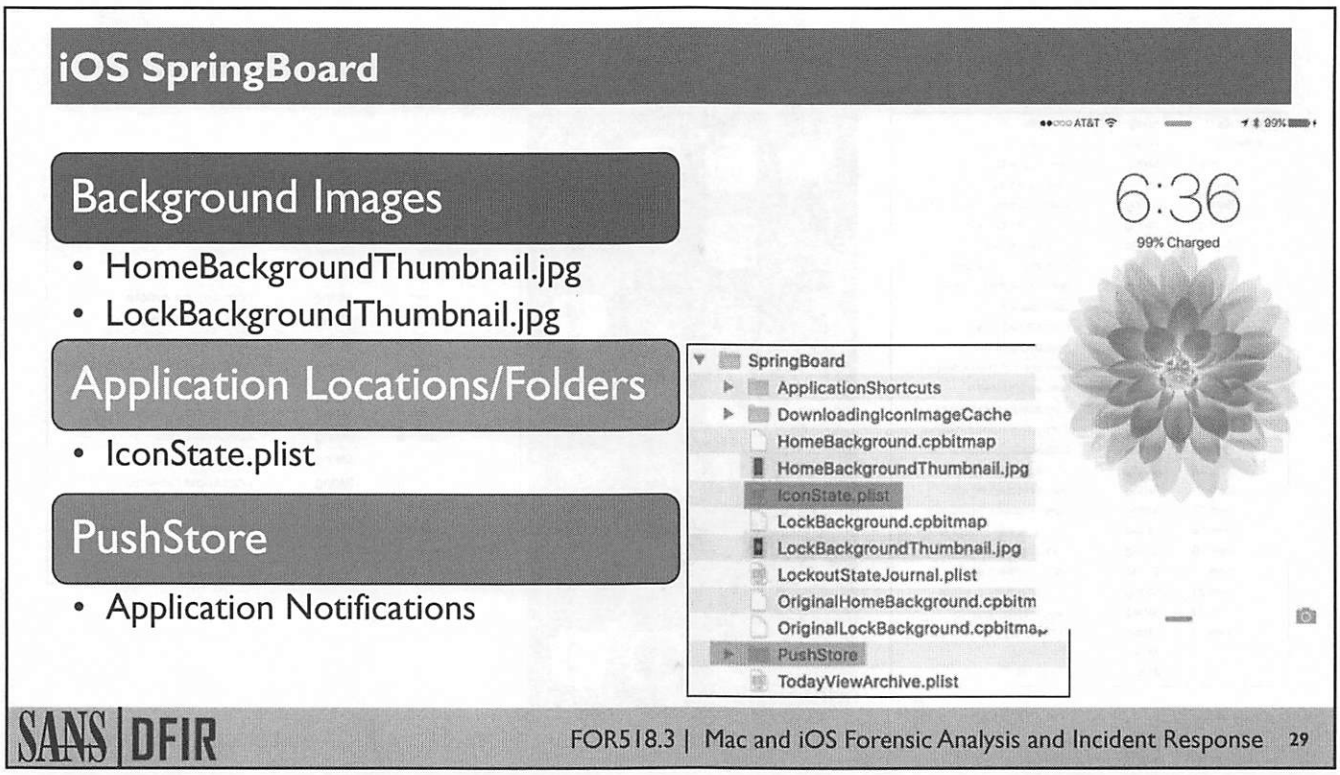
Key	Type	Value
BuildMachineOSBuild	String	11D17a
Localization native development region	String	English
Executable file	String	com.apple.qtkitserver
Bundle identifier	String	com.apple.qtkitserver
InfoDictionary version	String	6.0
Bundle name	String	com.apple.qtkitserver
Bundle OS Type code	String	XPC!
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
DTCompiler	String	
DTPlatformBuild	String	11D17a
DTPlatformVersion	String	GM
DTSDKBuild	String	11D17a
DTSDKName	String	
DTXcode	String	0410
DTXcodeBuild	String	11D17a
Application is agent (UIElement)	Boolean	YES
▽ XPCService	Diction...	(2 items)
▽ EnvironmentVariables	Diction...	(1 item)
MallocCorruptionAbort	String	1
ServiceType	String	Application



The Graphical User Interface for each operating system is named something different, and each works a bit differently because of the type of interaction from the user. For instance, macOS is meant to be used with Trackpad/Mouse/Keyboard, tvOS with a remote, and iOS and watchOS with your fingers!

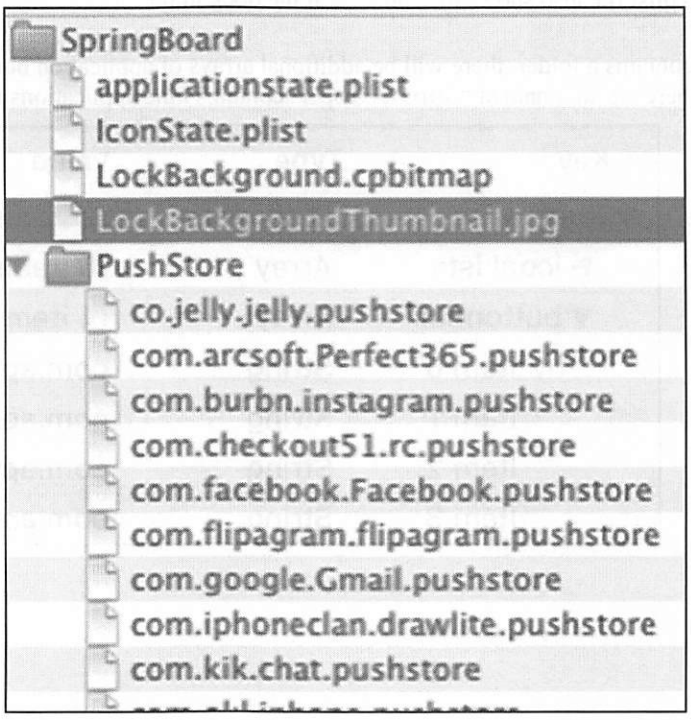
These interfaces can give us some detail into how a user uses their devices.

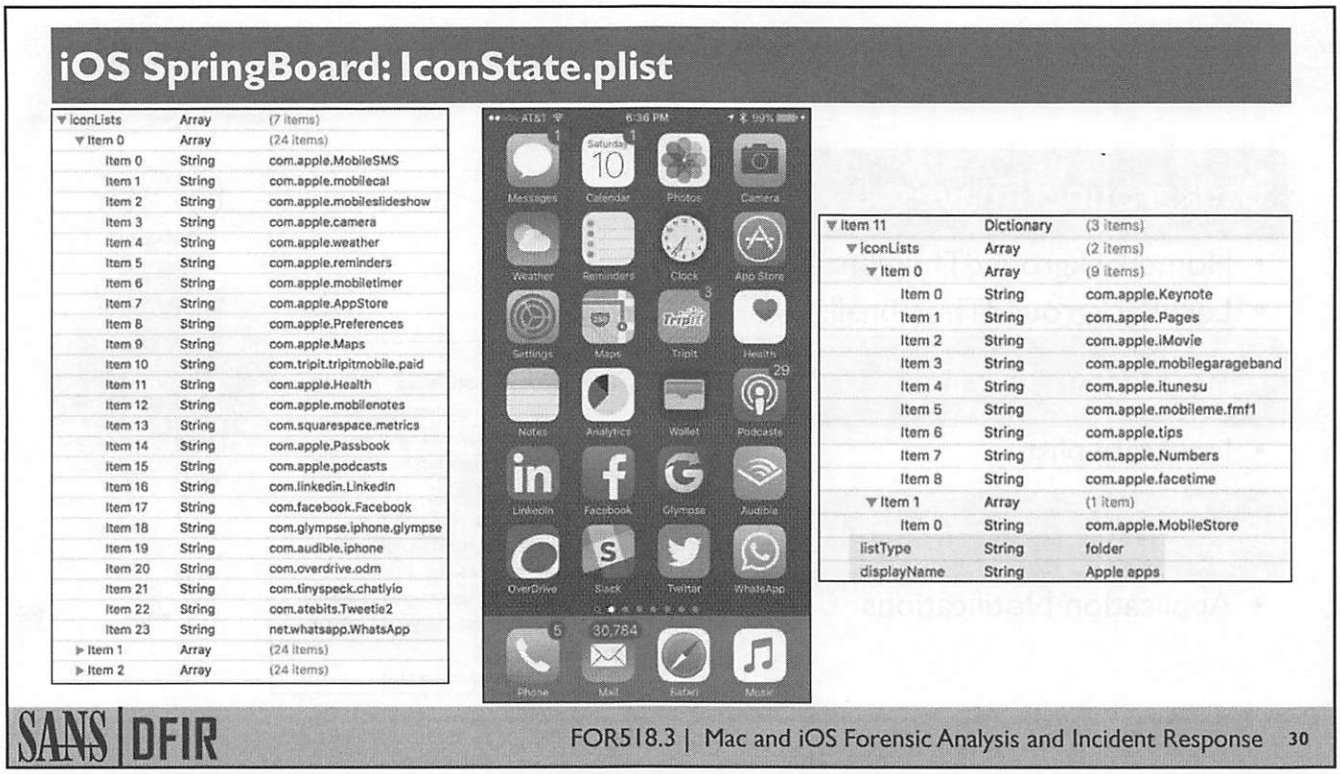
Reference:
`http://newosxbook.com/articles/TvOS.html`



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

The interface on iOS devices is called SpringBoard (in macOS, it is call `Finder`). The SpringBoard directory will contain interface-specific items, such as Home/Lock background pictures and the icon/screen layout, as well as notifications for various applications.





iconLists	Array	(7 items)
▼ Item 0	Array	(24 items)
Item 0	String	com.apple.MobileSMS
Item 1	String	com.apple.mobilecal
Item 2	String	com.apple.mobileslideshow
Item 3	String	com.apple.camera
Item 4	String	com.apple.weather
Item 5	String	com.apple.reminders
Item 6	String	com.apple.mobiletimer
Item 7	String	com.apple.AppStore
Item 8	String	com.apple.Preferences
Item 9	String	com.apple.Maps
Item 10	String	com.tripit.tripitmobile.paid
Item 11	String	com.apple.Health
Item 12	String	com.apple.mobilenotes
Item 13	String	com.squarespace.metrics
Item 14	String	com.apple.Passbook
Item 15	String	com.apple.podcasts
Item 16	String	com.linkedin.LinkedIn
Item 17	String	com.facebook.Facebook
Item 18	String	com.glympse.iphone.glympse
Item 19	String	com.audible.iphone
Item 20	String	com.overdrive.odm
Item 21	String	com.tinyspeck.chatlyio
Item 22	String	com.atebits.Tweetie2
Item 23	String	net.whatsapp.WhatsApp
▶ Item 1	Array	(24 items)
▶ Item 2	Array	(24 items)

Item 11	Dictionary	(3 items)
▼ iconLists	Array	(2 items)
▼ Item 0	Array	(9 items)
Item 0	String	com.apple.Keynote
Item 1	String	com.apple.Pages
Item 2	String	com.apple.iMovie
Item 3	String	com.apple.mobilegarageband
Item 4	String	com.apple.itunesu
Item 5	String	com.apple.mobileme.fmf1
Item 6	String	com.apple.tips
Item 7	String	com.apple.Numbers
Item 8	String	com.apple.facetime
▼ Item 1	Array	(1 item)
Item 0	String	com.apple.MobileStore
listType	String	folder
displayName	String	Apple apps

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

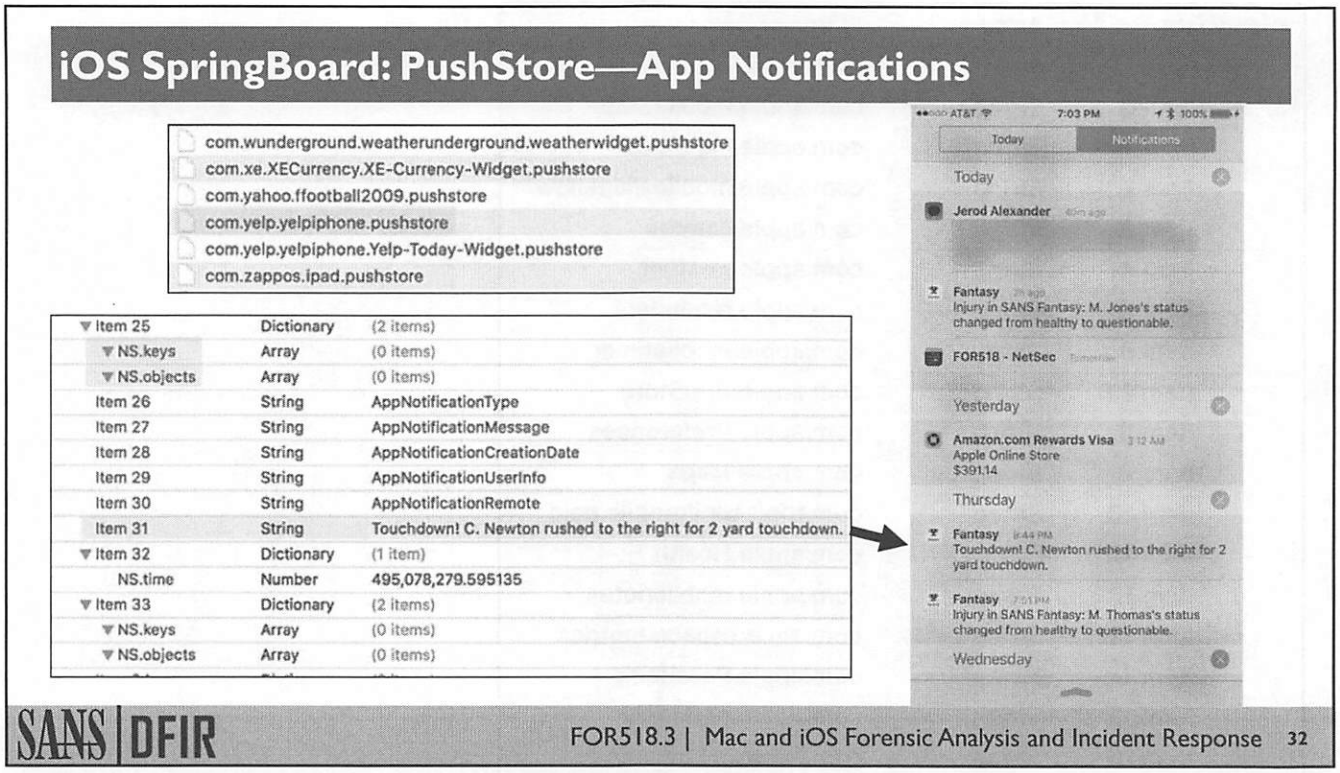
SpringBoard controls how each icon and folder is laid out on each screen. The `IconState.plist` file contains an "`Item`" for each screen under "`iconLists`". In the example above, "`Item 0`" is this device's first home screen. This plist file also shows the contents of the dock in the "`buttonBar`" key, shown below.

If a screen contains a folder, there will be additional arrays of application bundle ids, as shown in the screenshot to the right, where a folder named "`Apple apps`" contains nine applications in a folder.

Key	Type	Value
▼ Root	Dictionary	(2 items)
▶ iconLists	Array	(6 items)
▼ buttonBar	Array	(4 items)
Item 0	String	com.apple.mobilephone
Item 1	String	com.apple.mobilemail
Item 2	String	com.apple.mobilesafari
Item 3	String	com.apple.Music

▽ iconLists	Array	(7 items)
▽ Item 0	Array	(24 items)
Item 0	String	com.apple.MobileSMS
Item 1	String	com.apple.mobilecal
Item 2	String	com.apple.mobileslideshow
Item 3	String	com.apple.camera
Item 4	String	com.apple.weather
Item 5	String	com.apple.reminders
Item 6	String	com.apple.mobiletimer
Item 7	String	com.apple.AppStore
Item 8	String	com.apple.Preferences
Item 9	String	com.apple.Maps
Item 10	String	com.tripit.tripitmobile.paid
Item 11	String	com.apple.Health
Item 12	String	com.apple.mobilenotes
Item 13	String	com.squarespace.metrics
Item 14	String	com.apple.Passbook
Item 15	String	com.apple.podcasts
Item 16	String	com.linkedin.LinkedIn
Item 17	String	com.facebook.Facebook
Item 18	String	com.glympse.iphone.glympse
Item 19	String	com.audible.iphone
Item 20	String	com.overdrive.odm
Item 21	String	com.tinyspeck.chatlyio
Item 22	String	com.atebits.Tweetie2
Item 23	String	net.whatsapp.WhatsApp
▷ Item 1	Array	(24 items)
▷ Item 2	Array	

▽ Item 11	Dictionary	(3 items)
▽ iconLists	Array	(2 items)
▽ Item 0	Array	(9 items)
Item 0	String	com.apple.Keynote
Item 1	String	com.apple.Pages
Item 2	String	com.apple.iMovie
Item 3	String	com.apple.mobilegarageband
Item 4	String	com.apple.itunesu
Item 5	String	com.apple.mobileme.fmf1
Item 6	String	com.apple.tips
Item 7	String	com.apple.Numbers
Item 8	String	com.apple.facetime
▽ Item 1	Array	(1 item)
Item 0	String	com.apple.MobileStore
listType	String	folder
displayName	String	Apple apps



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

In the `PushStore` directory, a list of files named `<bundle_id>.pushstore` are stored for each application. Each `pushstore` file contains a specific application's notifications. These plist files are using the NSKeyedArchiver plist format.

A review of the contents of these plists can provide an insight into user activity for a particular application. The example above shows the pushstore data for the Yahoo Fantasy Football application.

While not shown in the screenshot, this pushstore file contained historical data from the previous season's Fantasy Football application usage.

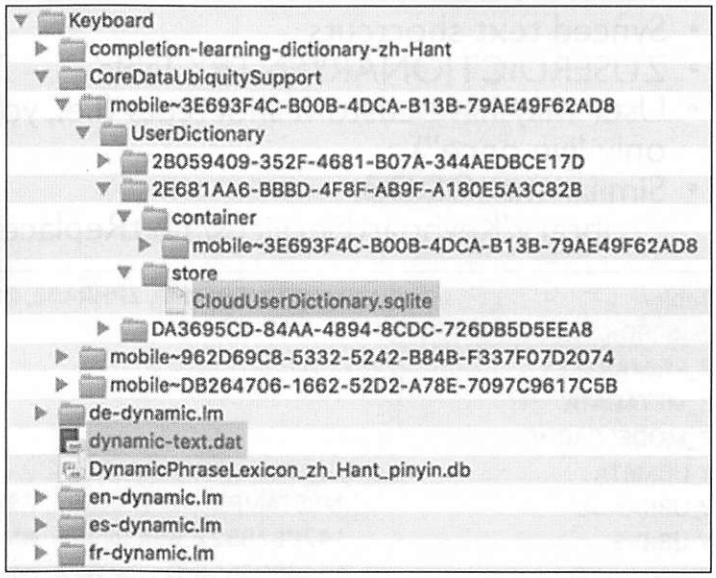
iOS Keyboard

Text Replacement/Shortcuts

- CloudUserDictionary.sqlite

Autocorrection and Predictive Text

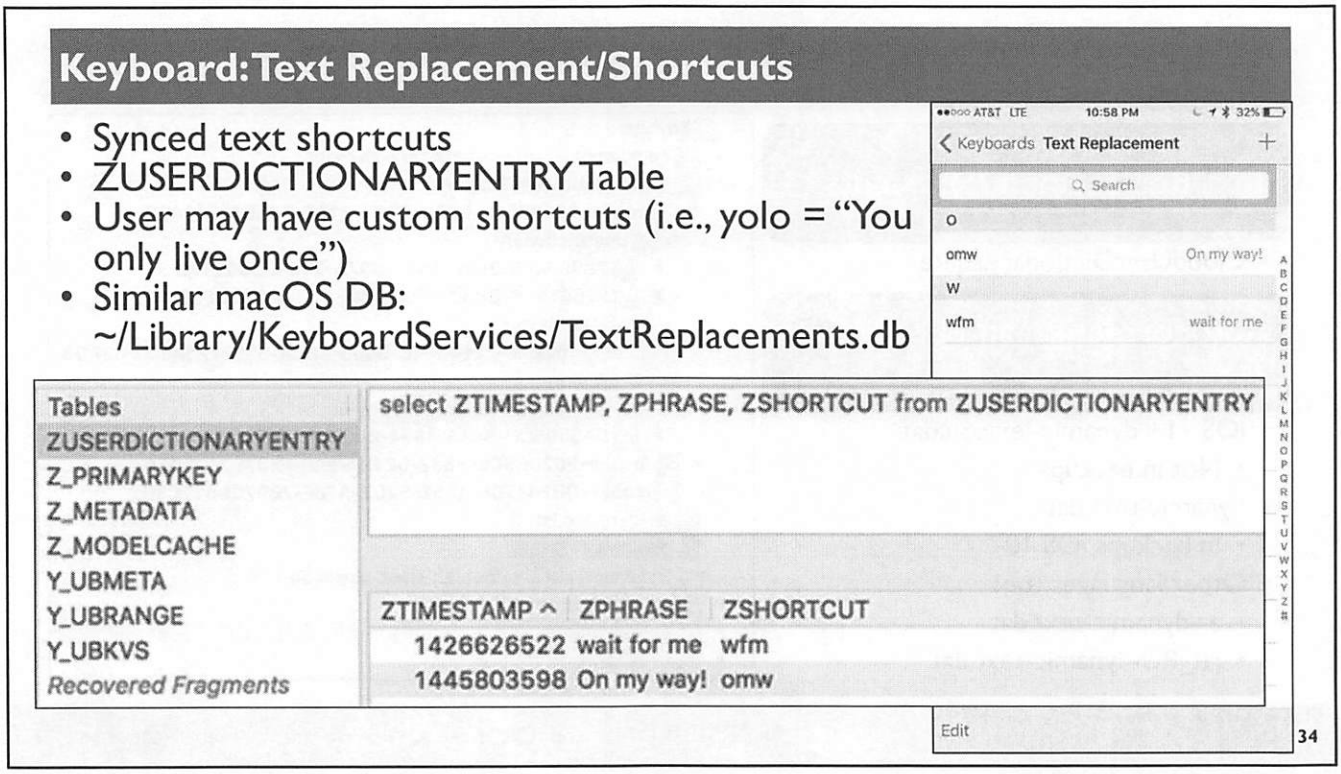
- iOS 11+ dynamic-lexicon.dat
  - Not in backups
- dynamic-text.dat
  - In backups iOS 10-
- Other languages too!
  - ar-dynamic-text.dat
  - ru_RU-dynamic-text.dat



Physical: `/private/var/mobile/Library/Keyboard/`
Backup/File System: `/mobile/Library/Keyboard/` or `/KeyboardDomain/Library/Keyboard/`

By default, on English-based devices, you will likely see at least two keyboards: English and Emoji. Users do have the ability to install additional keyboards. They may help them type in other languages, such as Arabic, Chinese, or Cyrillic characters. When users are using these keyboards, over time, certain words are recorded to help with autocorrection and predictive text. These words are stored in the `dynamic-text.dat` files. Note the default (English) is stored in `dynamic-text.dat`. Additional languages may have their own files (i.e., `ar-dynamic-text.dat` for Arabic or `ru_RU-dynamic-text.dat` for Russian Cyrillic).

Users may also want to configure their own text shortcuts. These are stored in the `CloudUserDictionary.sqlite` database files.

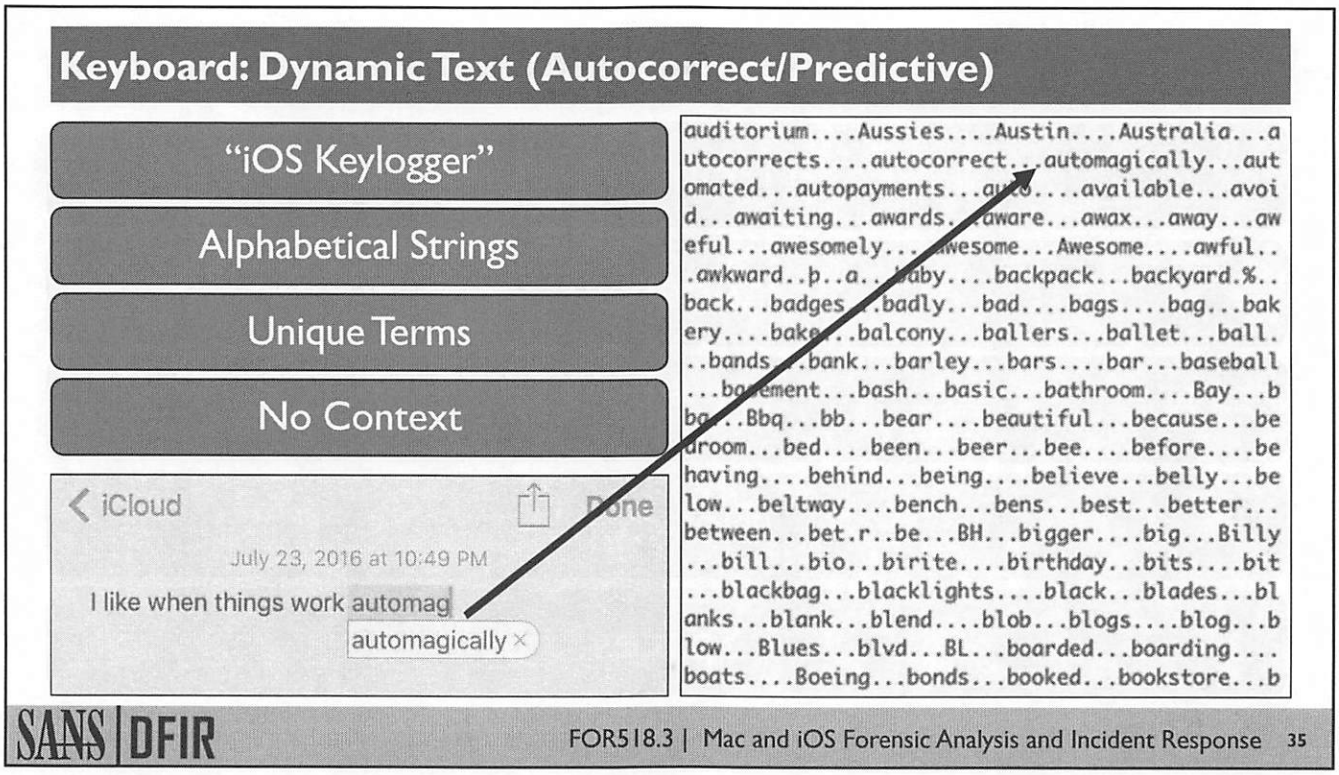


Physical: `/private/var/mobile/Library/Keyboard/`
Backup/File System: `/mobile/Library/Keyboard/` or `/KeyboardDomain/Library/Keyboard/`

Users may also want to configure their own text shortcuts. These are stored in the `CloudUserDictionary.sqlite` database files. These shortcuts may provide some insight into what the user regularly types.

For instance, instead of writing "`You only live once`", the user may just want to type "`yolo`" to make typing faster. When the user types "`yolo`", the actual text gets replaced with "`You only live once`".

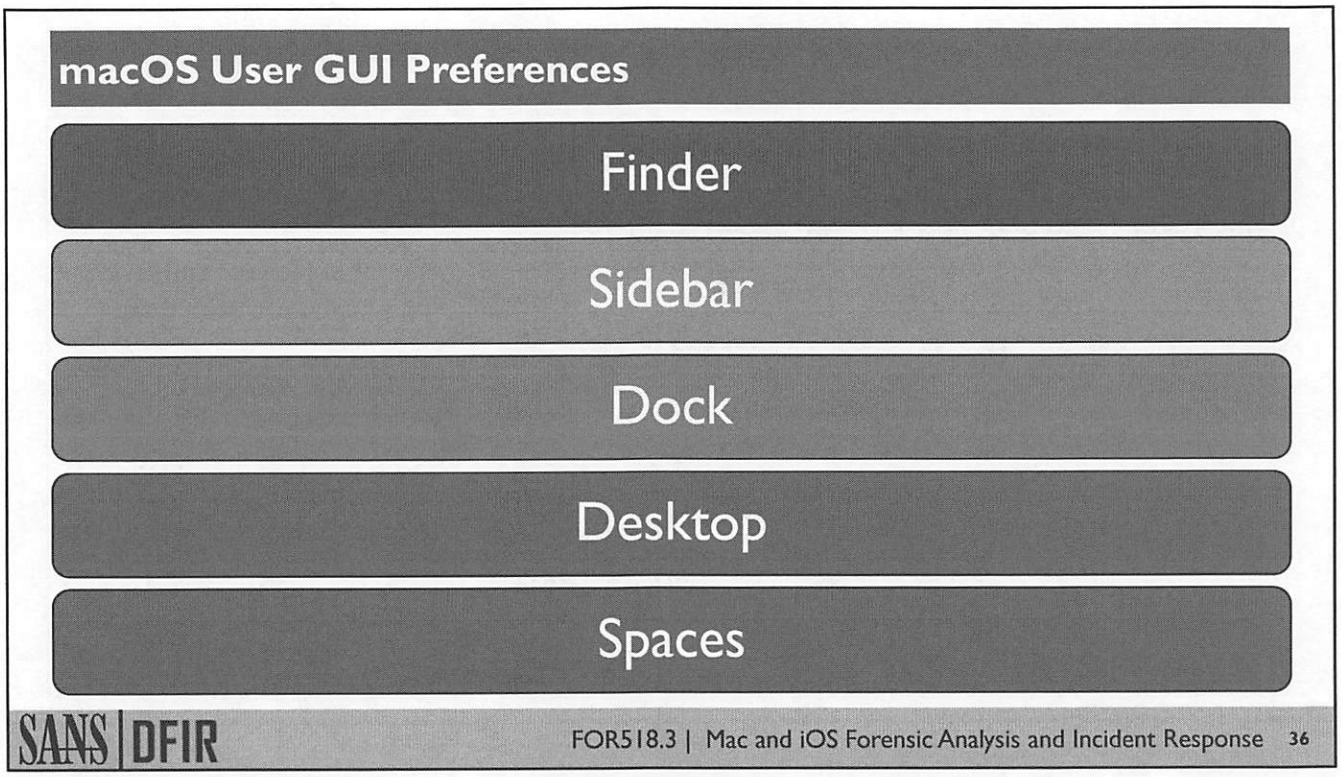
Similar database on macOS: `~/Library/KeyboardServices/TextReplacements.db`



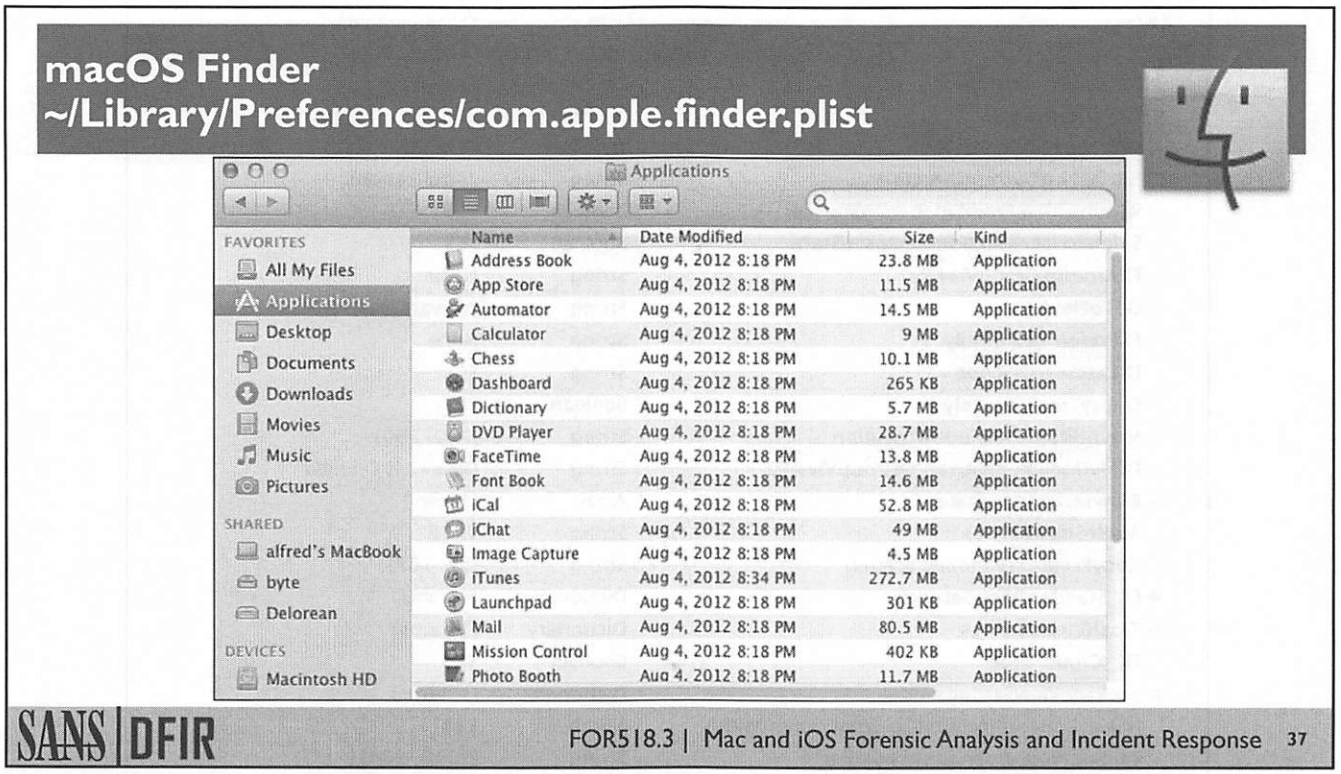
Physical: `/private/var/mobile/Library/Keyboard/`
Backup/File System: `/mobile/Library/Keyboard/` or `/KeyboardDomain/Library/Keyboard/`

These dynamic text files have been around for a long time; they are often known as the "iOS keylogger" files. They do not act as a true keylogger; however, a forensic analysis of these files may provide some insight into what the user likes to type. These strings may include names, nicknames, places, slang, etc.

In theory, these files should not contain anything sensitive such as usernames or passwords. Items typed into "secure" web forms or login areas do not get recorded in this file. This, however, does not stop the user from typing sensitive information in nonsecure areas, such as the Notes application.



A user's workspace can say many things about a person: What applications are likely to be used more often, how organized the user is, or what directories are important to them. The OS X interface has many entities that can give you a peek into how a person uses their computer.



The Apple Finder is comparable to the Windows Explorer. It is the main application allowing access to applications, files, directories, networks, and other media.

On the left sidebar, the Finder categories are shown. These, of course, are configurable by the user.

- Favorites: By default, the user directories are shown, such as Documents, Pictures, Music, Downloads, etc.
- Shared: These are the other devices on the network than can be used for file storage, other computers that can be "screen shared", etc.
- Devices: Contains the volumes the system has access to: The host volume (Macintosh HD), USB drives, CDs/DVDs, FireWire, Thunderbolt, mounted DMGs, etc.

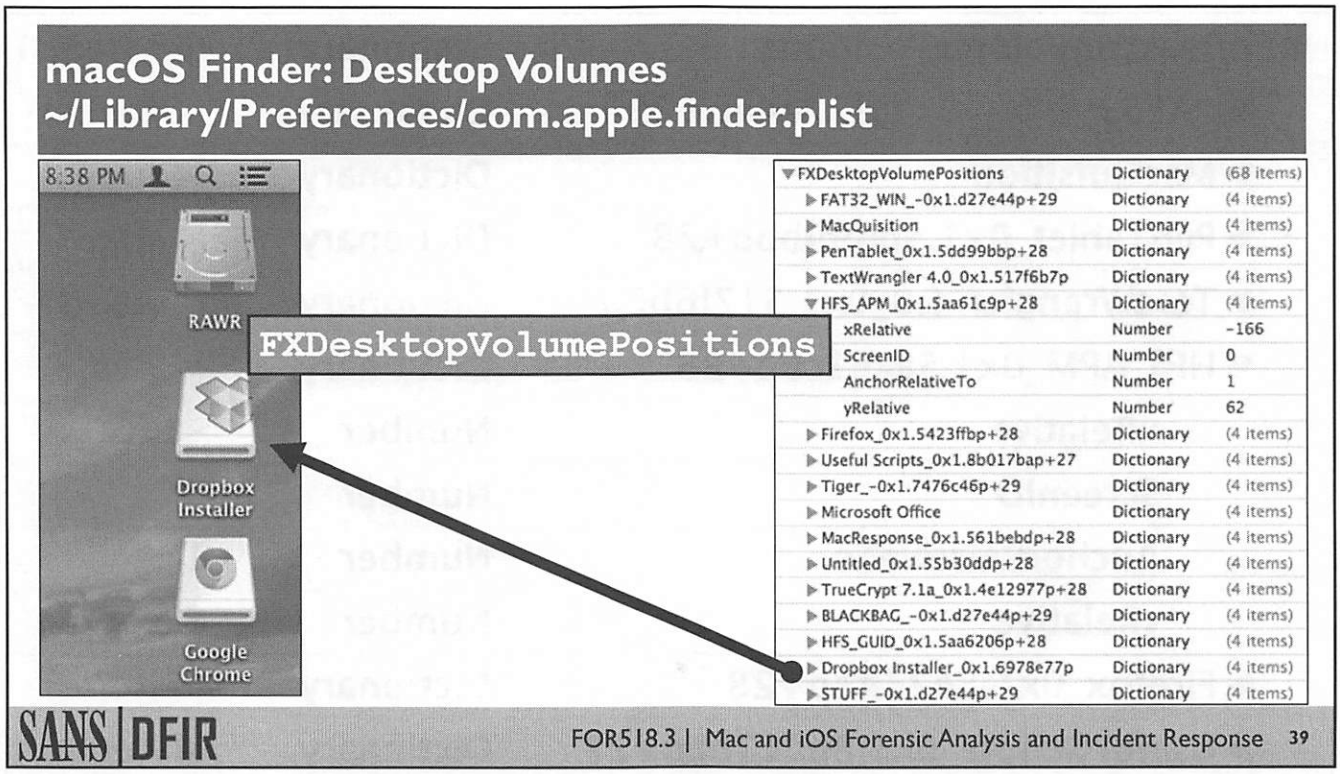
Among the preference keys that will be covered, the `com.apple.finder.plist` contains a huge amount of user preference data. Almost every configurable item of the Finder can be found in this plist.

For example:

- Show Mounted Servers
- Show Mounted Hard Drives
- Column Preference
- Empty Trash Securely
- X and Y Coordinates of different GUI features

Note: Each version of OS X may have different property list keys, though the notable ones are the same. 10.6 does not have nearly as many configuration details as systems 10.7+ (see the next page).

Key	Type	Value
▼ Root	Dictionary	(56 items)
▶ NetworkViewSettings	Dictionary	(1 item)
FlowViewHeight	Number	255
FXPreferredViewStyle	String	Nlsv
NSNavLastCurrentDirectory	String	/Applications
RemoveIDiskFromSidebarOnStartup	Boolean	YES
SidebarPlacesSectionDisclosedState	Boolean	YES
FXArrangeGroupViewBy	String	Name
GoToField	String	/private/var/log/
FXPreferredGroupBy	String	None
FXLastSearchScope	String	SCcf
EmptyTrashSecurely	Boolean	YES
MountProgressWindowLocation	String	{760, 290}
FXMyDocumentsArrangeGroupViewBy	String	Date Last Opened
▶ BrowserWindowState	Array	(1 item)
AppleShowAllFiles	String	TRUE
CopyProgressWindowLocation	String	{57, 167}
▶ FK_StandardViewSettings	Dictionary	(4 items)
▶ TrashViewSettings	Dictionary	(3 items)
FK_ICloudMode	Boolean	YES
▶ PackageViewSettings	Dictionary	(3 items)
NSNavBrowserPreferedColumnContentWidth	Number	186
BackupProgressWindowLocation	String	{1130, 1041}
▶ NSToolbar Configuration Browser	Dictionary	(7 items)
FXConnectToBounds	String	{{717, 621}, {486, 231}}
▶ RecentSearches	Array	(3 items)
FXToolbarUpgradedToTenEight	Number	1
FXPreferredSearchViewStyleVersion	String	%00%00%00%01
FXConnectToLastURL	String	afp://somethingelse
▶ StandardViewSettings	Dictionary	(4 items)
▶ DesktopViewSettings	Dictionary	(2 items)
EmptyTrashProgressWindowLocation	String	{760, 260}
NewWindowTargetPath	String	file:///.file/id=6562758.335740/
ShowRemovableMediaOnDesktop	Boolean	YES
NewWindowTarget	String	PfHm
NSNavLastRootDirectory	String	/Applications
PreferencesWindow.LastSelection	String	SDBR
SidebarDevicesSectionDisclosedState	Boolean	YES
SearchMyDocsLibraryBrowseWithCustomViewStyle	Boolean	NO
▶ FXDesktopVolumePositions	Dictionary	(18 items)
▶ FXInfoPanesExpanded	Dictionary	(4 items)
LastTrashState	Boolean	YES
ShowHardDrivesOnDesktop	Boolean	YES
FK_LinenViewStyle	String	icnv
▶ ComputerViewSettings	Dictionary	(4 items)
DownloadsFolderListViewSettingsVersion	Number	1
FK_SidebarWidth	Number	192
SidebarSharedSectionDisclosedState	Boolean	YES
FXToolbarUpgradedToTenSeven	Number	1
ShowExternalHardDrivesOnDesktop	Boolean	YES
ShowMountedServersOnDesktop	Boolean	YES
▶ SearchViewSettings	Dictionary	(3 items)
FXPreferencesWindow.Location	String	{{479, 191}, {355, 559}}
▶ FXRecentFolders	Array	(7 items)
FXPreferredSearchViewStyle	String	Nlsv



The Finder application stores the mounted volumes in the `com.apple.finder.plist`. (Remember we already discussed how this plist also stores the `Recent Folders` in the `FXRecentFolders` key.)

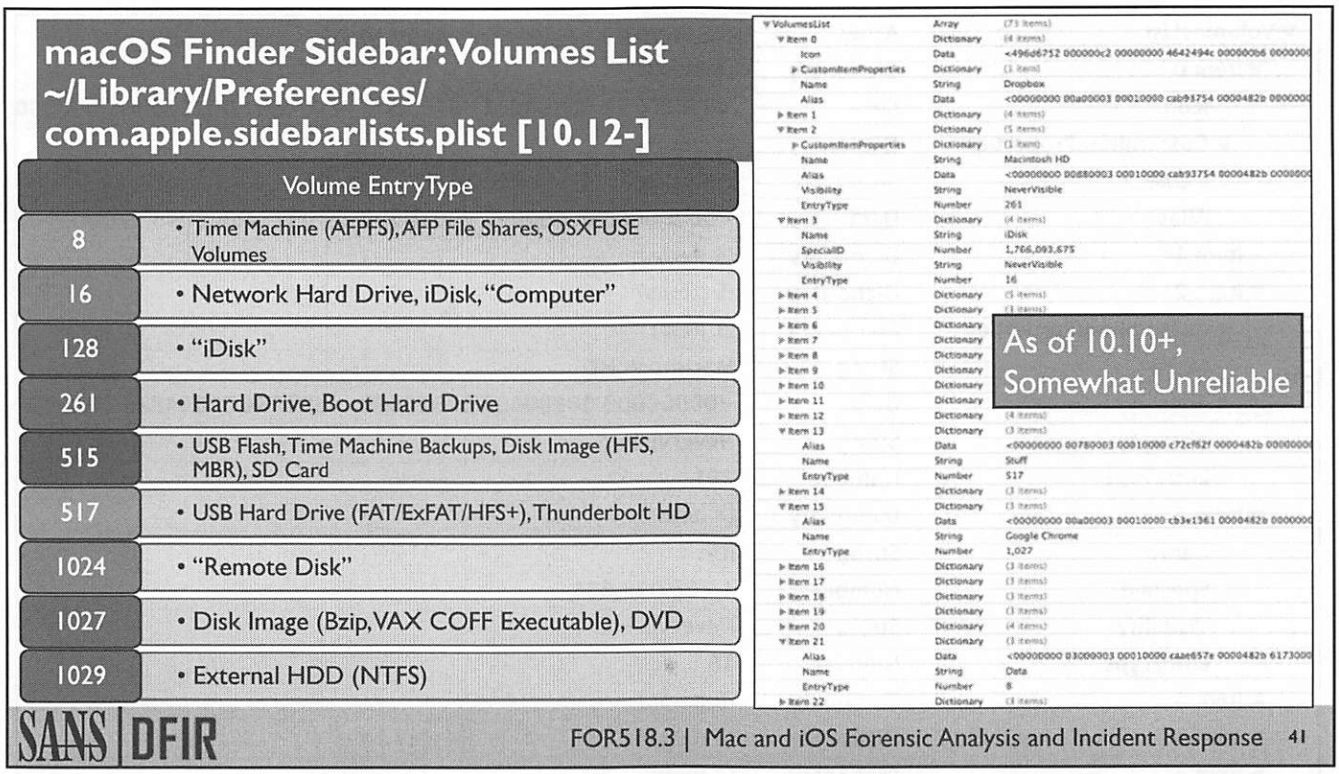
The `FXDesktopVolumePositions` key mainly stores the `X` and `Y` coordinates of volumes when mounted on the Desktop. This key can be useful for forensic analysts because it stores all the volumes that were mounted by the user. It unfortunately does not store the date when it was last mounted, but we may be able to determine that with analysis of the logs. Knowing when a volume is mounted can help an investigator determine when it was in use.

Each volume is listed by its mounted volume name, and some entries have an appended number.* It is unknown what this number represents.

Note: If the user does not have the Finder preferences configured to show items on the desktop, this key will not exist.

* Darren Freestone has determined this number is a "hexadecimal floating point constant" value that stores the volume creation timestamp of HFS+ volumes but only a negative value for FAT/ExFAT volumes.

▼ FXDesktopVolumePositions	Dictionary	(68 items)
▶ FAT32_WIN_-0x1.d27e44p+29	Dictionary	(4 items)
▶ MacQuisition	Dictionary	(4 items)
▶ PenTablet_0x1.5dd99bbp+28	Dictionary	(4 items)
▶ TextWrangler 4.0_0x1.517f6b7p	Dictionary	(4 items)
▼ HFS_APM_0x1.5aa61c9p+28	Dictionary	(4 items)
xRelative	Number	-166
ScreenID	Number	0
AnchorRelativeTo	Number	1
yRelative	Number	62
▶ Firefox_0x1.5423ffbp+28	Dictionary	(4 items)
▶ Useful Scripts_0x1.8b017bap+27	Dictionary	(4 items)
▶ Tiger_-0x1.7476c46p+29	Dictionary	(4 items)
▶ Microsoft Office	Dictionary	(4 items)
▶ MacResponse_0x1.561bebdp+28	Dictionary	(4 items)
▶ Untitled_0x1.55b30ddp+28	Dictionary	(4 items)
▶ TrueCrypt 7.1a_0x1.4e12977p+28	Dictionary	(4 items)
▶ BLACKBAG_-0x1.d27e44p+29	Dictionary	(4 items)
▶ HFS_GUID_0x1.5aa6206p+28	Dictionary	(4 items)
▶ Dropbox Installer_0x1.6978e77p	Dictionary	(4 items)
▶ STUFF_-0x1.d27e44p+29	Dictionary	(4 items)



The `com.apple.sidebarlists.plist` contains the items found in the sidebar of the Finder. The `system items` or `favorites` key (version dependent) contains configurable items, such as:

- Show Removable Drives (USB, External HDDs)
- Show Hard Drives
- Show Ejectables (iDevices, CDs, DVDs)
- Show Servers

As of 10.10, the author has found this plist to be somewhat unreliable. It will show some volumes, but certainly not all volumes mounted on the system. The `VolumesList` keys contain the items found in the sidebar of the Finder. This key may contain a large history of items related to mounted volumes.

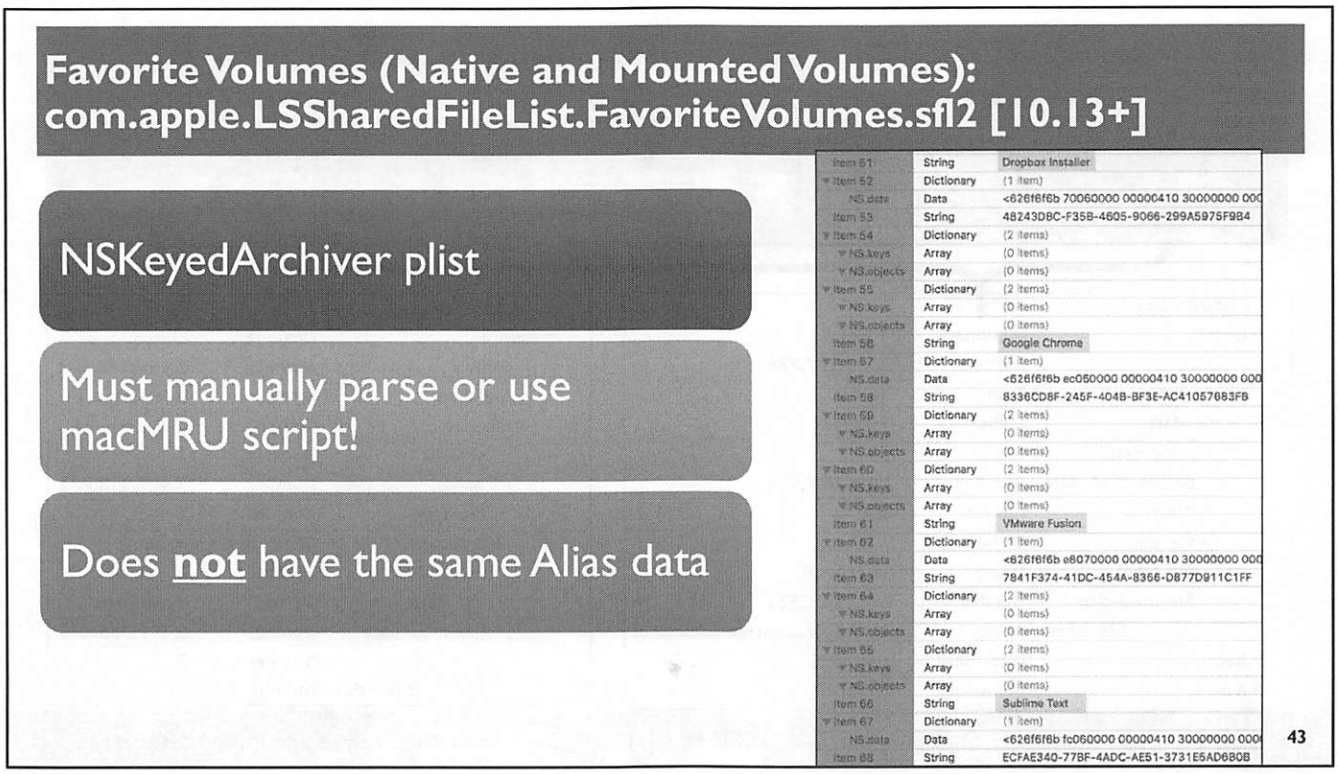
Each entry may have:

- `Alias Data`
- `Name`: Volume Name
- `Icon Data`: If the volume uses a particular icon.
- `Visibility Setting`: Whether configured to be viewable by the user or not.
- `EntryType`: Type of disk; some of the types are listed in the table above.

The `EntryType` keys may have different meanings, depending on the version of OS X you are analyzing. This plist does not appear to get created on 10.13 systems.

▼ favorites	Dictionary	(7 items)
▼ CustomListProperties	Dictionary	(2 items)
com.apple.LSSharedFileList.VolumesListMigrated	Boolean	YES
com.apple.LSSharedFileList.Restricted.upgraded	Boolean	YES
ShowRemovable	Boolean	YES
ShowHardDisks	Boolean	YES
ShowEjectables	Boolean	YES
▶ VolumesList	Array	(73 items)
ShowServers	Boolean	YES
Controller	String	VolumesList

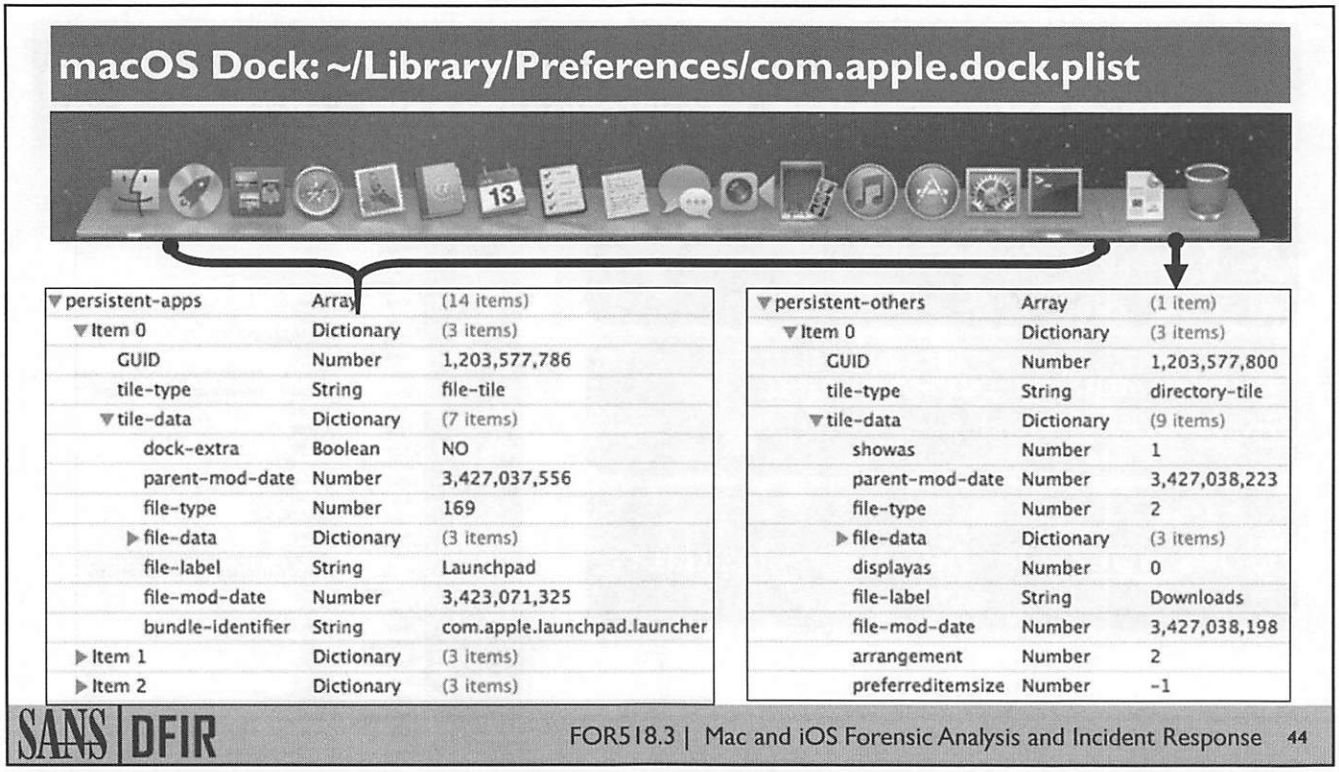
▼VolumesList	Array	(73 items)
▼Item 0	Dictionary	(4 items)
Icon	Data	<496d6752 000000c2 00000000 4642494c 000000b6 0000000
▶CustomItemProperties	Dictionary	(1 item)
Name	String	Dropbox
Alias	Data	<00000000 00a00003 00010000 cab93754 0000482b 0000000
▶Item 1	Dictionary	(4 items)
▼Item 2	Dictionary	(5 items)
▶CustomItemProperties	Dictionary	(1 item)
Name	String	Macintosh HD
Alias	Data	<00000000 00880003 00010000 cab93754 0000482b 0000000
Visibility	String	NeverVisible
EntryType	Number	261
▼Item 3	Dictionary	(4 items)
Name	String	iDisk
SpecialID	Number	1,766,093,675
Visibility	String	NeverVisible
EntryType	Number	16
▶Item 4	Dictionary	(5 items)
▶Item 5	Dictionary	(3 items)
▶Item 6	Dictionary	(4 items)
▶Item 7	Dictionary	(4 items)
▶Item 8	Dictionary	(4 items)
▶Item 9	Dictionary	(4 items)
▶Item 10	Dictionary	(4 items)
▶Item 11	Dictionary	(4 items)
▶Item 12	Dictionary	(4 items)
▼Item 13	Dictionary	(3 items)
Alias	Data	<00000000 00780003 00010000 c72cf62f 0000482b 0000000
Name	String	Stuff
EntryType	Number	517
▶Item 14	Dictionary	(3 items)
▼Item 15	Dictionary	(3 items)
Alias	Data	<00000000 00a00003 00010000 cb3e1361 0000482b 0000000
Name	String	Google Chrome
EntryType	Number	1,027
▶Item 16	Dictionary	(3 items)
▶Item 17	Dictionary	(3 items)
▶Item 18	Dictionary	(3 items)
▶Item 19	Dictionary	(3 items)
▶Item 20	Dictionary	(4 items)
▼Item 21	Dictionary	(3 items)
Alias	Data	<00000000 03000003 00010000 caae657e 0000482b 6173000
Name	String	Data
EntryType	Number	8
▶Item 22	Dictionary	(3 items)



In 10.13, `com.apple.sidebarlists.plist` no longer exists. Instead, most of the volume information is found in the `com.apple.LSSharedFileList.FavoriteVolumes.sfl2` plist file in the `~/Library/Application Support/com.apple.sharedfilelist/` directory.

This is an NSKeyedArchiver plist file, which means manual parsing will likely need to be performed in order to understand the contents and context of this data. The example output below is the output from the macMRU.py script shown earlier in the Mac MRU module.

```
[Item Number: 4 | (UUID:'48243DBC-F35B-4605-9066-299A5975F9B4') | Visibility: 0] Name: 'Dropbox Installer'
    CustomItemProperties:
      NSURLVolumeIsInternalKey: False
      NSURLVolumeIsEjectableKey: False
      com.apple.LSSharedFileList.VolumeGroup: 3
      NSURLVolumeIsRootFileSystemKey: False
      NSURLIsVolumeKey: True

[Item Number: 5 | (UUID:'8336CD8F-245F-404B-BF3E-AC41057683FB') | Visibility: 0] Name: 'Google Chrome'
    CustomItemProperties:
      NSURLVolumeIsInternalKey: False
      NSURLVolumeIsEjectableKey: False
      com.apple.LSSharedFileList.VolumeGroup: 3
      NSURLVolumeIsRootFileSystemKey: False
      NSURLIsVolumeKey: True

[Item Number: 6 | (UUID:'7841F374-41DC-454A-8356-D877D911C1FF') | Visibility: 0] Name: 'VMware Fusion'
    CustomItemProperties:
      NSURLVolumeIsInternalKey: False
      NSURLVolumeIsEjectableKey: False
      com.apple.LSSharedFileList.VolumeGroup: 3
      NSURLVolumeIsRootFileSystemKey: False
      NSURLIsVolumeKey: True

[Item Number: 7 | (UUID:'ECFAE340-77BF-4ADC-AE51-3731E5AD6B0B') | Visibility: 0] Name: 'Sublime Text'
    CustomItemProperties:
      NSURLVolumeIsInternalKey: False
      NSURLVolumeIsEjectableKey: False
      com.apple.LSSharedFileList.VolumeGroup: 3
      NSURLVolumeIsRootFileSystemKey: False
      NSURLIsVolumeKey: True
```



The dock can be used to determine what applications a user is more likely to use often.

The `com.apple.dock.plist` contains two main keys:

- `persistent-apps`: These are the applications on the left of the dock separator (not including Finder). These applications are listed in order from left to right (`Item 0, Item 1, Item 2 ... Item N`).
- `persistent-others`: These are folders that can be docked and viewed—the default folder in the dock is Download; however, many times you'll also see the Document directory.

Each Item has data about the application or directory included:

- View Types
- File Data, including Alias data and file path
- Dock Label
- Modification Dates (File and Parent)

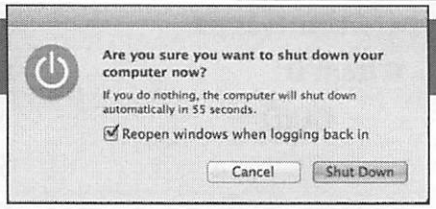
Other preference items may be found in this property list:

- Dock Auto Hide
- Full Trash
- Icon Sizes
- Dock Magnification

▼ persistent-apps	Array	(14 items)
▼ Item 0	Dictionary	(3 items)
GUID	Number	1,203,577,786
tile-type	String	file-tile
▼ tile-data	Dictionary	(7 items)
dock-extra	Boolean	NO
parent-mod-date	Number	3,427,037,556
file-type	Number	169
▶ file-data	Dictionary	(3 items)
file-label	String	Launchpad
file-mod-date	Number	3,423,071,325
bundle-identifier	String	com.apple.launchpad.launcher
▶ Item 1	Dictionary	(3 items)
▶ Item 2	Dictionary	(3 items)

▼ persistent-others	Array	(1 item)
▼ Item 0	Dictionary	(3 items)
GUID	Number	1,203,577,800
tile-type	String	directory-tile
▼ tile-data	Dictionary	(9 items)
showas	Number	1
parent-mod-date	Number	3,427,038,223
file-type	Number	2
▶ file-data	Dictionary	(3 items)
displayas	Number	0
file-label	String	Downloads
file-mod-date	Number	3,427,038,198
arrangement	Number	2
preferreditemsize	Number	-1

Saved Application State



Introduced in 10.7 as "resume" feature

- Launches applications and windows to the previous state, after a system reboot or an exited application.

Locations:

- macOS Legacy: ~/Library/Saved Application State/
- macOS Sandbox: ~/Library/Containers/<Bundle ID>/Data/Library/Application Support/<App Name>/Saved Application State
- iOS: <Application Directory>/Library/Saved Application State/<bundle_id>.savedState/

```
nibble:Saved Application State sledwards$ pwd
/Users/sledwards/Library/Saved Application State
nibble:Saved Application State sledwards$ ls -lt
total 32
drwx------  7 sledwards  staff  238 Nov 17 16:03 com.apple.finder.savedState
drwx------  5 sledwards  staff  170 Nov 17 15:46 com.suavetech.0xED.savedState
drwx------  6 sledwards  staff  204 Nov 17 15:44 com.apple.iChat.savedState
drwx------  6 sledwards  staff  204 Nov 17 15:36 com.apple.dt.Xcode.savedState
drwx------  5 sledwards  staff  170 Nov 17 15:17 com.apple.systempreferences.savedState
drwx------  6 sledwards  staff  204 Nov 17 14:57 com.realmacsoftware.littlesnapper.savedState
drwx------  8 sledwards  staff  272 Nov 17 13:40 com.apple.Terminal.savedState
drwx------  6 sledwards  staff  204 Nov 17 11:08 com.microsoft.Powerpoint.savedState
drwx------  6 sledwards  staff  204 Nov 17 10:43 com.apple.appstore.savedState
drwx------  5 sledwards  staff  170 Nov 17 10:31 com.apple.installer.savedState
drwx------  5 sledwards  staff  170 Nov 17 09:53 com.microsoft.autoupdate2.savedState
drwx------@ 5 sledwards  staff  170 Nov 17 09:46 com.apple.Safari.savedState
drwx------  5 sledwards  staff  170 Nov 15 03:46 com.apple.iCal.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:51 com.apple.iTunes.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:51 com.apple.keychainaccess.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:51 com.vmware.fusion.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:51 net.sourceforge.sqlitebrowser.savedState
drwx------@ 5 sledwards  staff  170 Nov 14 05:48 com.google.Chrome.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:45 com.apple.ActivityMonitor.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:45 com.apple.Console.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:45 com.ridiculousfish.HexFiend.savedState
drwx------  5 sledwards  staff  170 Nov 14 05:45 com.apple.Stickies.savedState
lrwxr-xr-x  1 sledwards  staff  113 Nov 10 09:37 com.apple.mail.savedState -> /Users/sledwards/L
ibrary/Containers/com.apple.mail/Data/Library/Saved Application State/com.apple.mail.savedState
lrwxr-xr-x  1 sledwards  staff  123 Nov 10 09:37 com.apple.reminders.savedState -> /Users/sledwa
rds/Library/Containers/com.apple.reminders/Data/Library/Saved Application State/com.apple.remind
ers.savedState
lrwxr-xr-x  1 sledwards  staff  115 Nov 10 09:37 com.apple.Notes.savedState -> /Users/sledwards/
Library/Containers/com.apple.Notes/Data/Library/Saved Application State/com.apple.Notes.savedSta
te
```


The Saved Application State, introduced in 10.7, is used to return applications to their previous state after a reboot. When shutting down, a user is given a choice to "`reopen windows when logging back in`".

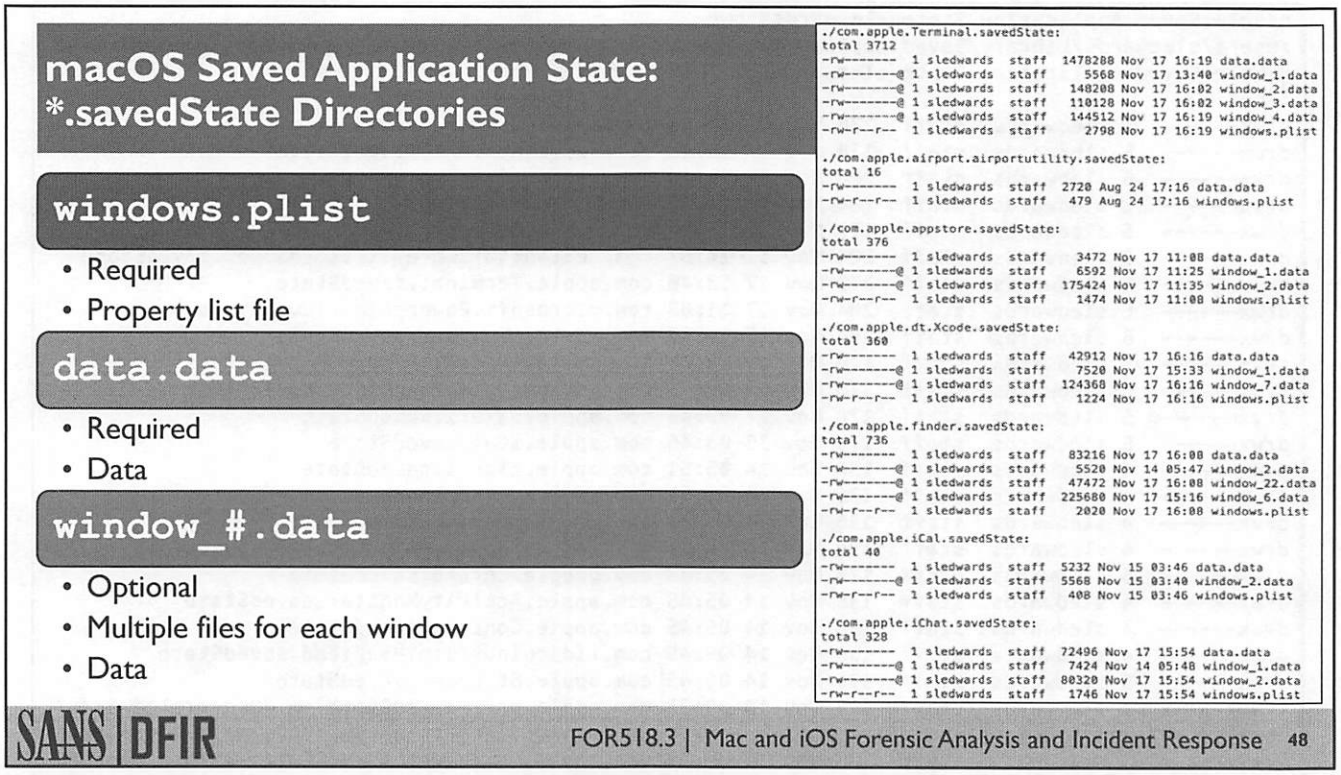
The directory `~/Library/Saved Application State/` also contains links to the `.savedState` directories in the sandboxed application containers.

On iOS, this is used to return to a previous application "tab" within that application.

Each application has its own directory named `<Bundle ID>.savedState`. Notice the links to the application sandbox containers.

The existence of these directories means the user has used these applications.

```
nibble:Saved Application State sledwards$ pwd
/Users/sledwards/Library/Saved Application State
nibble:Saved Application State sledwards$ ls -lt
total 32
drwx------  7 sledwards  staff  238 Nov 17 16:03 com.apple.finder.savedState
drwx------  5 sledwards  staff  170 Nov 17 15:46 com.suavetech.0xED.savedState
drwx------  6 sledwards  staff  204 Nov 17 15:44 com.apple.iChat.savedState
drwx------  6 sledwards  staff  204 Nov 17 15:36 com.apple.dt.Xcode.savedState
drwx------  5 sledwards  staff  170 Nov 17 15:17 com.apple.systempreferences.savedState
drwx------  6 sledwards  staff  204 Nov 17 14:57 com.realmacsoftware.littlesnapper.savedState
drwx------  8 sledwards  staff  272 Nov 17 13:40 com.apple.Terminal.savedState
drwx------  6 sledwards  staff  204 Nov 17 11:08 com.microsoft.Powerpoint.savedState
drwx------  6 sledwards  staff  204 Nov 17 10:43 com.apple.appstore.savedState
drwx------  5 sledwards  staff  170 Nov 17 10:31 com.apple.installer.savedState
drwx------  5 sledwards  staff  170 Nov 17 09:53 com.microsoft.autoupdate2.savedState
drwx------@ 5 sledwards  staff  170 Nov 17 09:46 com.apple.Safari.savedState
drwx------  5 sledwards  staff  170 Nov 15 03:46 com.apple.iCal.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:51 com.apple.iTunes.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:51 com.apple.keychainaccess.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:51 com.vmware.fusion.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:51 net.sourceforge.sqlitebrowser.savedState
drwx------@ 5 sledwards  staff  170 Nov 14 05:48 com.google.Chrome.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:45 com.apple.ActivityMonitor.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:45 com.apple.Console.savedState
drwx------  4 sledwards  staff  136 Nov 14 05:45 com.ridiculousfish.HexFiend.savedState
drwx------  5 sledwards  staff  170 Nov 14 05:45 com.apple.Stickies.savedState
lrwxr-xr-x  1 sledwards  staff  113 Nov 10 09:37 com.apple.mail.savedState -> /Users/sledwards/L
ibrary/Containers/com.apple.mail/Data/Library/Saved Application State/com.apple.mail.savedState
lrwxr-xr-x  1 sledwards  staff  123 Nov 10 09:37 com.apple.reminders.savedState -> /Users/sledwa
rds/Library/Containers/com.apple.reminders/Data/Library/Saved Application State/com.apple.remind
ers.savedState
lrwxr-xr-x  1 sledwards  staff  115 Nov 10 09:37 com.apple.Notes.savedState -> /Users/sledwards/
Library/Containers/com.apple.Notes/Data/Library/Saved Application State/com.apple.Notes.savedSta
te
```

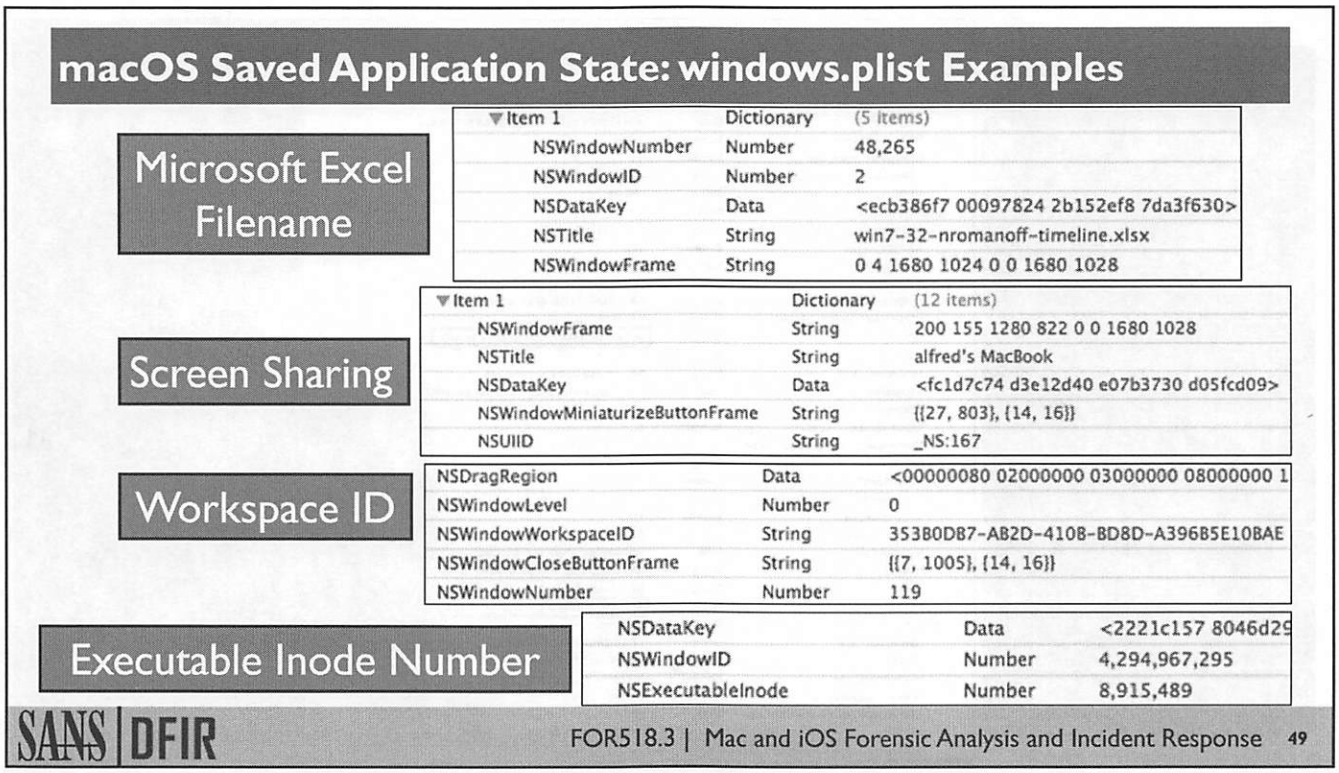


Each `.savedState` directory contains at least two files:

- `windows.plist`
- `data.data`

Additional files named `windows_1.data`, `windows_2.data`, etc. may also be created.

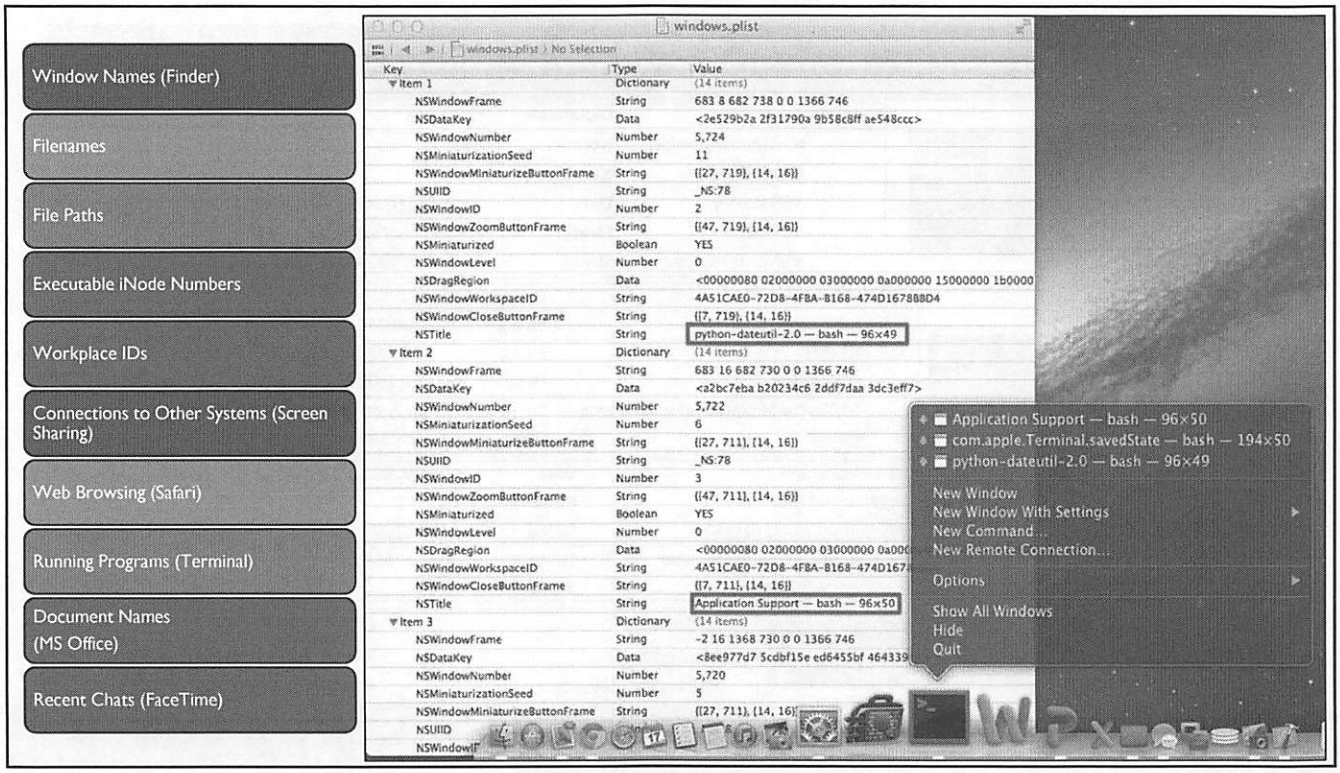
On iOS, these files are named "`restorationinfo.plist`" and "`data.data`". Depending on the acquisition and iOS version, the `data.data` may be unencrypted.



The screenshots show examples of what can be found in Saved Application State `windows.plist` files.

From top to bottom:

- A filename used with Microsoft Excel
- A connected Screen Sharing System: `alfred's MacBook`
- A Workspace ID used in Spaces
- An `NSExecutableInode` number that can be tracked back to a specific application



The `windows.plist` file contains information about the application windows that should be "reopened".

While there are some windows positioning data, the interesting part is the `NSTitle` for each Item. This contains the name of the application window. In the screenshot, the Terminal window names are shown in the right-click menu. This information may help a forensic analyst determine what was running or otherwise happening with that application.

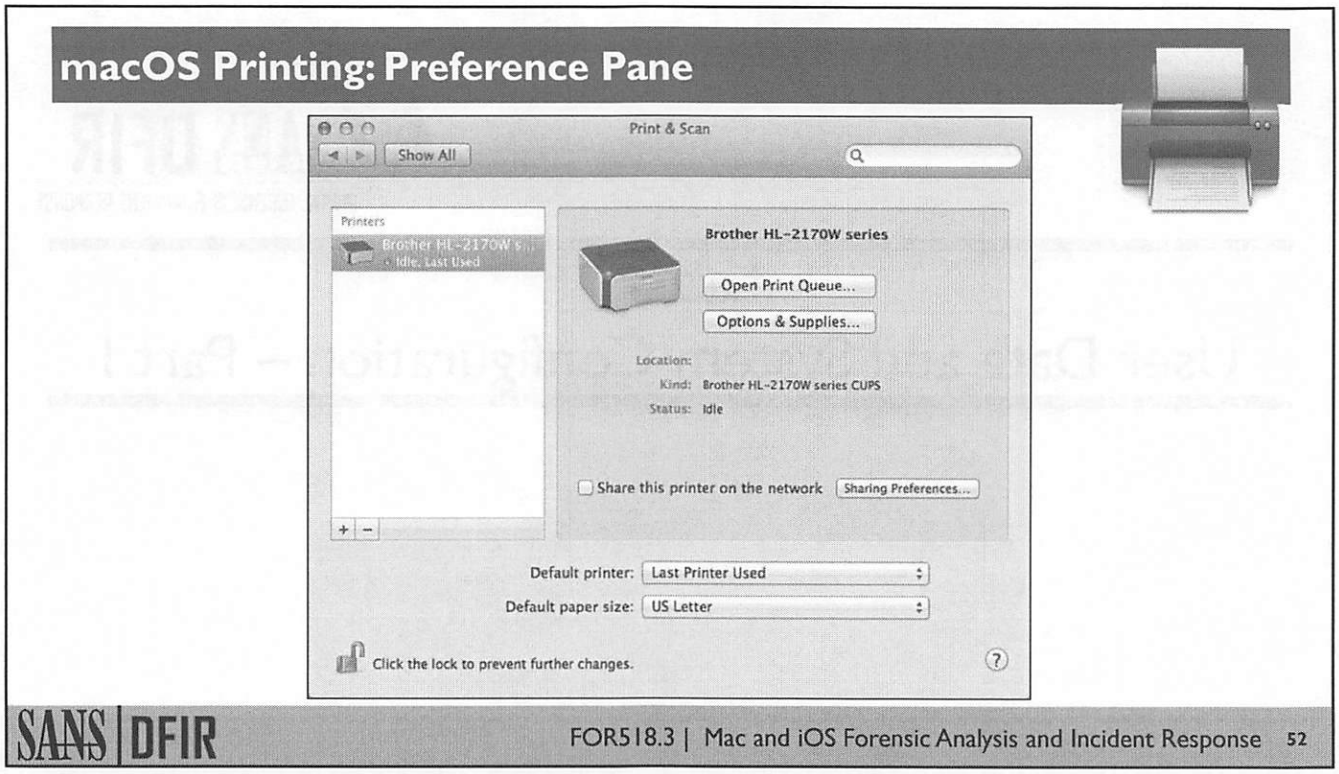
Lots of good forensic tidbits can be found looking at the `windows.plist` file for an application.

- Finder holds the names of opened Windows.
- Filenames are often found, specifically with applications that open documents. Microsoft Office will have the names of opened files because they use the filename as the window name.
- Workplace IDs, used to keep track of Spaces, can be used to determine if an application is opened on a specific Space.
- Screen Sharing will save the names of connected systems.
- Browser programs like Safari save the webpage title as the window name.
- Programs running in Terminal can be seen in window names; look for things like `sudo`, `ssh`, etc.

Many `window.plist` files have a key named `NSExecutableInode`. This is used to store the iNode number of the related application executable.

Lab 3.1
User Data and System Configuration – Part 1

This page intentionally left blank.



The printing preference pane contains printer settings for the system. The screenshot shows one printer, a Brother HL-2170W, set up on the system.

macOS Printing: /Library/Preferences/org.cups.printers.plist

▼ Root	Array	(2 items)
▼ Item 0	Dictionary	(9 items)
printer-name	String	Brother_HL_2170W_series
printer-info	String	Brother HL-2170W series
printer-is-accepting-jobs	Boolean	YES
printer-location	String	
printer-make-and-model	String	Brother HL-2170W series CUPS
printer-state	Number	3
▶ printer-state-reasons	Array	(0 items)
printer-type	Number	8,433,732
device-uri	String	dnssd://Brother%20HL-2170W%20series._pdl-datastream._tcp.local./?bidi
▼ Item 1	Dictionary	(9 items)
printer-name	String	Brother_HL_5140_series
printer-info	String	Brother HL-5140 series
printer-is-accepting-jobs	Boolean	YES
printer-location	String	word
printer-make-and-model	String	Brother HL-5140 series CUPS
printer-state	Number	3
▶ printer-state-reasons	Array	(1 item)
printer-type	Number	45,124
device-uri	String	usb://Brother/HL-5140%20series?serial=J4J541146

The `org.cups.printers.plist` file located in `/Library/Preferences/` contains details about installed printers. Each printer will be under its own `Item` key.

The screenshot above shows two printers. The Brother HL-2170W was accessed via the network, while the Brother HL-5140 was accessed via a USB cable.

Another configuration file, the `printers.conf` file located in `/etc/cups/`, contains specific printer information for each printer installed on the system. The screenshot shows the Brother printer is shared among other system users ("`Shared Yes`").

More printer configurations can be found in a printer-specific configuration file similar to `Brother_HL_2170W_series.ppd` located in the `/etc/cups/ppd/` directory. A PostScript Printer Description (PPD) file contains the printer's specific capabilities, such as page size, resolution, color, and fonts.

The `StateTime` and `Attribute marker-change-time` timestamps are Unix epoch timestamps that show when the printer was configured on the system.

```
# Printer configuration file for CUPS v1.6.2
# Written by cupsd on 2013-05-24 08:22
# DO NOT EDIT THIS FILE WHEN CUPSD IS RUNNING
<Printer Brother_HL_2170W_series>
UUID urn:uuid:d0de313e-f1d1-3dc9-4e35-3453705570a1
Info Brother HL-2170W series
MakeModel Brother HL-2170W series CUPS
DeviceURI dnssd://Brother%20HL-2170W%20series._pdl-datastream._tcp.local./?bidi
State Idle
StateTime 1369355085
Type 8433732
Accepting Yes
Shared Yes
JobSheets none none
QuotaPeriod 0
PageLimit 0
KLimit 0
OpPolicy default
ErrorPolicy stop-printer
Attribute marker-colors \#000000
Attribute marker-levels -3
Attribute marker-names Black
Attribute marker-types toner
Attribute marker-change-time 1350222390
</Printer>
```

macOS Printer Control Files: /private/var/spool/cups

- Nine printer control jobs (c#######)
- Just use `strings`!

```
sh-3.2# pwd
/private/var/spool/cups
sh-3.2# ls -la
total 72
drwx--x---  13 root  _lp     442 Jun 14 10:36 .
drwxr-xr-x   7 root  wheel   238 May  9 19:22 ..
-rw-------   1 root  _lp    1841 May 31 20:26 c00001
-rw-------   1 root  _lp    1883 May 31 20:27 c00002
-rw-------   1 root  _lp    1883 Jun  1 17:26 c00003
-rw-------   1 root  _lp    1883 Jun  1 17:31 c00004
-rw-------   1 root  _lp    1815 Jun  1 17:28 c00005
-rw-------   1 root  _lp     723 Jun  1 17:31 c00006
-rw-------   1 root  _lp    1883 Jun  1 17:33 c00007
-rw-------   1 root  _lp    1883 Jun  1 17:38 c00008
-rw-------   1 root  _lp    1873 Jun 14 10:36 c00009
drwxrwxr-x   8 root  _lp     272 Jun 14 10:46 cache
drwxrwx--T   2 root  _lp      68 Jun 19  2011 tmp
```

```
bash-3.2# strings c00004
attributes-charset
utf-8H
attributes-natural-language
en-us
printer-uri
4ipp://localhost:631/printers/Brother_HL_2170W_seriesB
job-originating-user-name
oompaB
job-name
27-Day Forecast for Latit...d Longitude 77.36&deg;WB
AP_ColorMatchingMode
AP_ApplicationColorMatchingB
AP_D_InputSlot
collate
ColorModel
GrayB
,com.apple.print.DocumentTicket.PMSpoolFormat
application/pdfB
)com.apple.print.JobInfo.PMApplicationName
ChromeB
!com.apple.print.JobInfo.PMJobName
27-Day Forecast for Latit...d Longitude 77.36&deg;WB
"com.apple.print.JobInfo.PMJobOwner
oompaB
```

SANS | DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 55

Each print job has a printer control file located in the `/private/var/spool/cups/` directory. Each file labeled `c#####` correlates with the job ID found in the `page_log`. In the example shown above, the system has printed nine documents (`c00001-c00009`). Each control file contains print job metadata.

Each printer control file contains metadata about each print job, including which printer was used, originating user account, job name, and which application was used to print from. This information is stored in a proprietary format that is easily viewed using the strings command (although some binary characters could be mistaken for ASCII—watch out for this).

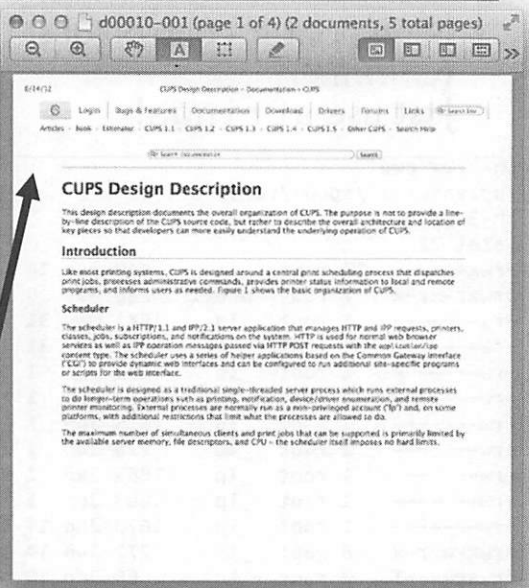
In the example above (Note: Some characters may be superfluous when using strings utility):

- The printer is located on the local network and is a Brother HL-2170W
- The originating user account is "oompa"
- The name of the job is "7-Day Forecast for Latit…d Longitude 77.36°W"
- The application used to print is the Chrome web browser

macOS Printer Data Files: /private/var/spool/cups

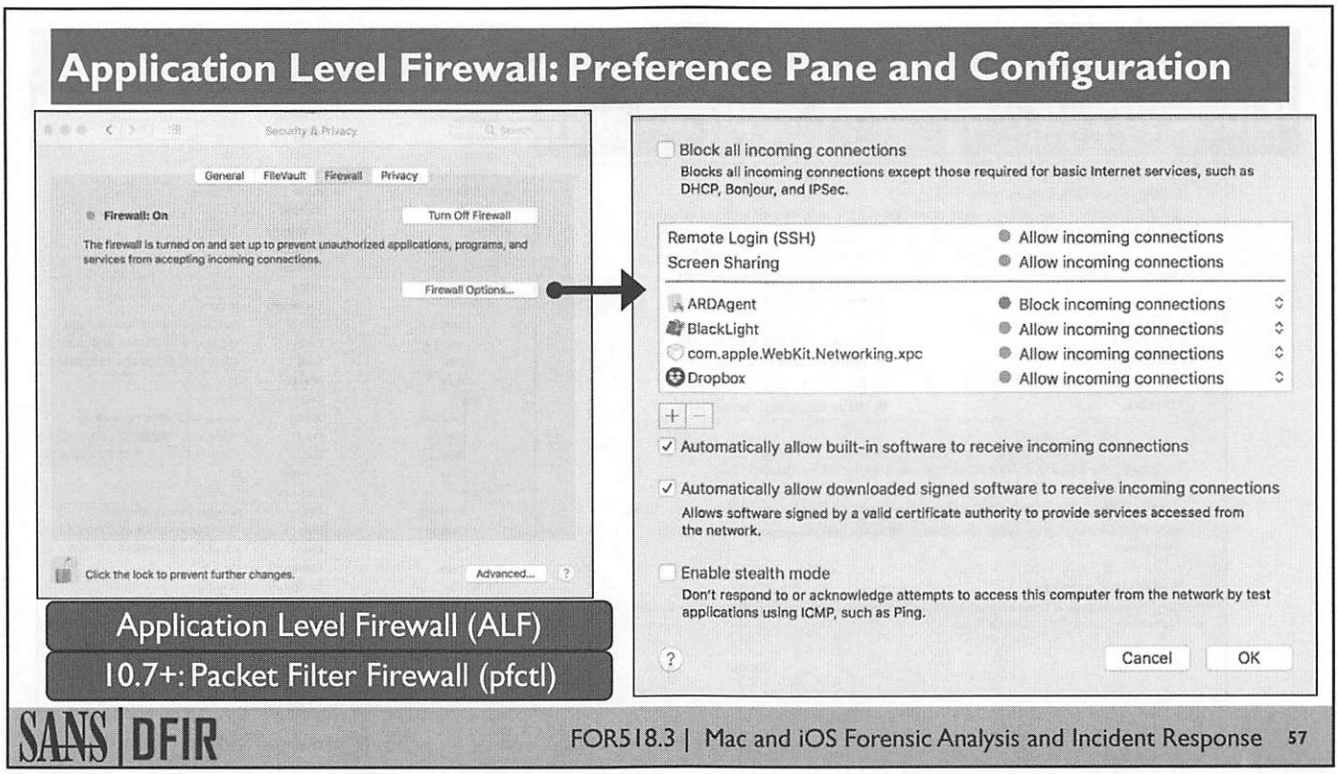
- Data files (d#######)
- Removed immediately after successful print
- PDF files

```
drwx--x---  15 root  _lp       510 Jun 14 12:36 .
drwxr-xr-x   7 root  wheel      238 May  9 19:22 ..
-rw-------   1 root  _lp      1841 May 31 20:26 c00001
-rw-------   1 root  _lp      1883 May 31 20:27 c00002
-rw-------   1 root  _lp      1883 Jun  1 17:26 c00003
-rw-------   1 root  _lp      1883 Jun  1 17:31 c00004
-rw-------   1 root  _lp      1815 Jun  1 17:28 c00005
-rw-------   1 root  _lp       723 Jun  1 17:31 c00006
-rw-------   1 root  _lp      1883 Jun  1 17:33 c00007
-rw-------   1 root  _lp      1883 Jun  1 17:38 c00008
-rw-------   1 root  _lp      1873 Jun 14 10:36 c00009
-rw-------   1 root  _lp      1878 Jun 14 12:36 c00010
drwxrwxr-x   8 root  _lp       272 Jun 14 12:36 cache
-rw-r-----   1 root  _lp    608373 Jun 14 12:35 d00010-001
drwxrwx--T   2 root  _lp        68 Jun 19  2011 tmp
```



Each printer control file has a matching printer data file; while the printer control files are persistent, the printer data files are not. These are created at the time of printing and are PDF files. Each is named in line with the printer control jobs (i.e., Printer control file `c00010` would have the data file `d00010-001`).

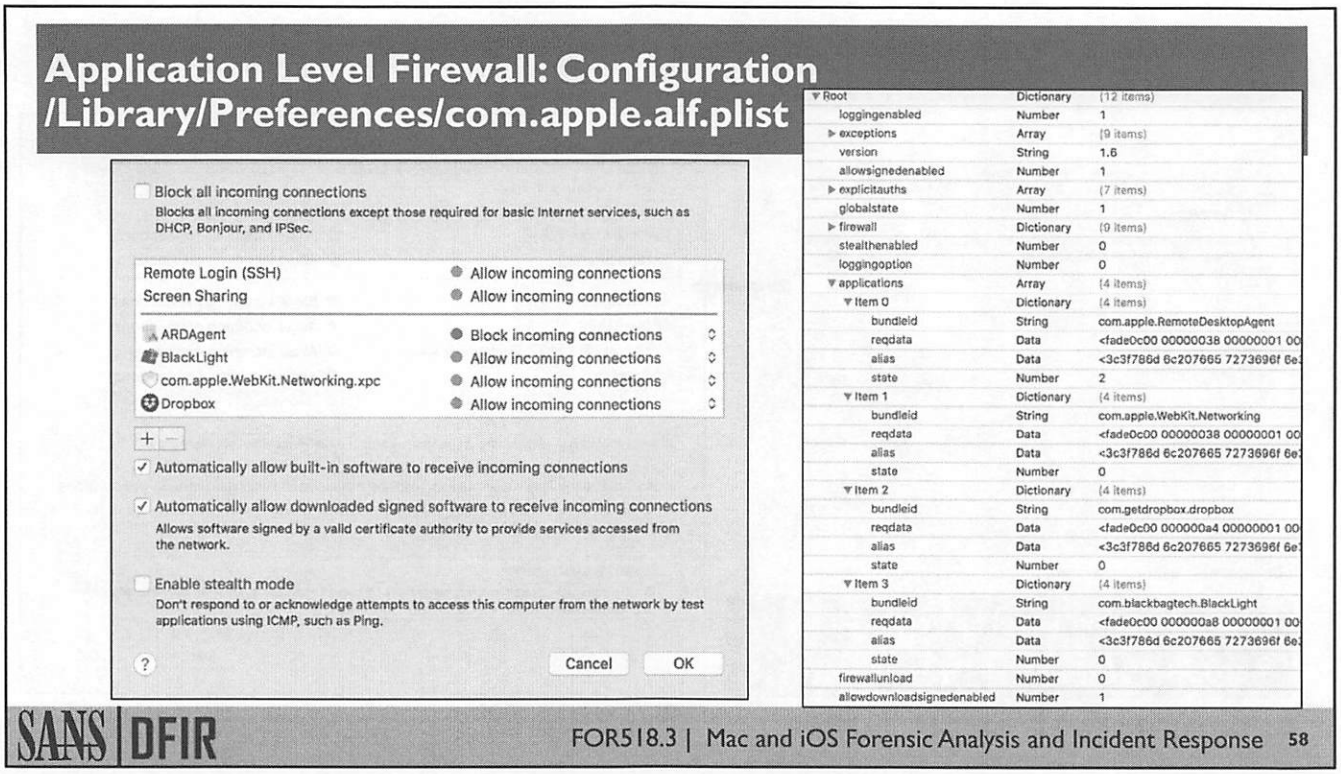
These files should be removed immediately after a print job has successfully printed. The files may be recoverable by carving out PDF files from the disk. If the print job has been canceled or otherwise produced an error, the files may persist for a while longer.



The default firewall program on macOS is the application level firewall (ALF), which can be configured in the `Security & Privacy` preferences panel under System Preferences. The firewall is not turned on by default on a newly installed system.

The screenshot on the left shows the `Firewall` tab on the `Security & Privacy` preferences pane. The screenshot on the right shows the `Firewall Options,` which show the remote access options and specific application configurations.

macOS systems come with two firewalls: An application level firewall (host-based application firewall) and an IP/Packet Filtering firewall. A non-savvy user may use the application level firewall via the GUI, while a more technical user may use the IP firewall via the `pfctl` command. If used, the Packet Filtering firewall configuration can be found in `/etc/pf.conf`.



The `com.apple.alf.plist` property file in the `/Library/Preferences` directory contains the configuration for the application level firewall.

The `globalstate` key determines if the firewall is enabled.

- 1 = Firewall Enabled
- 0 = Firewall Disabled

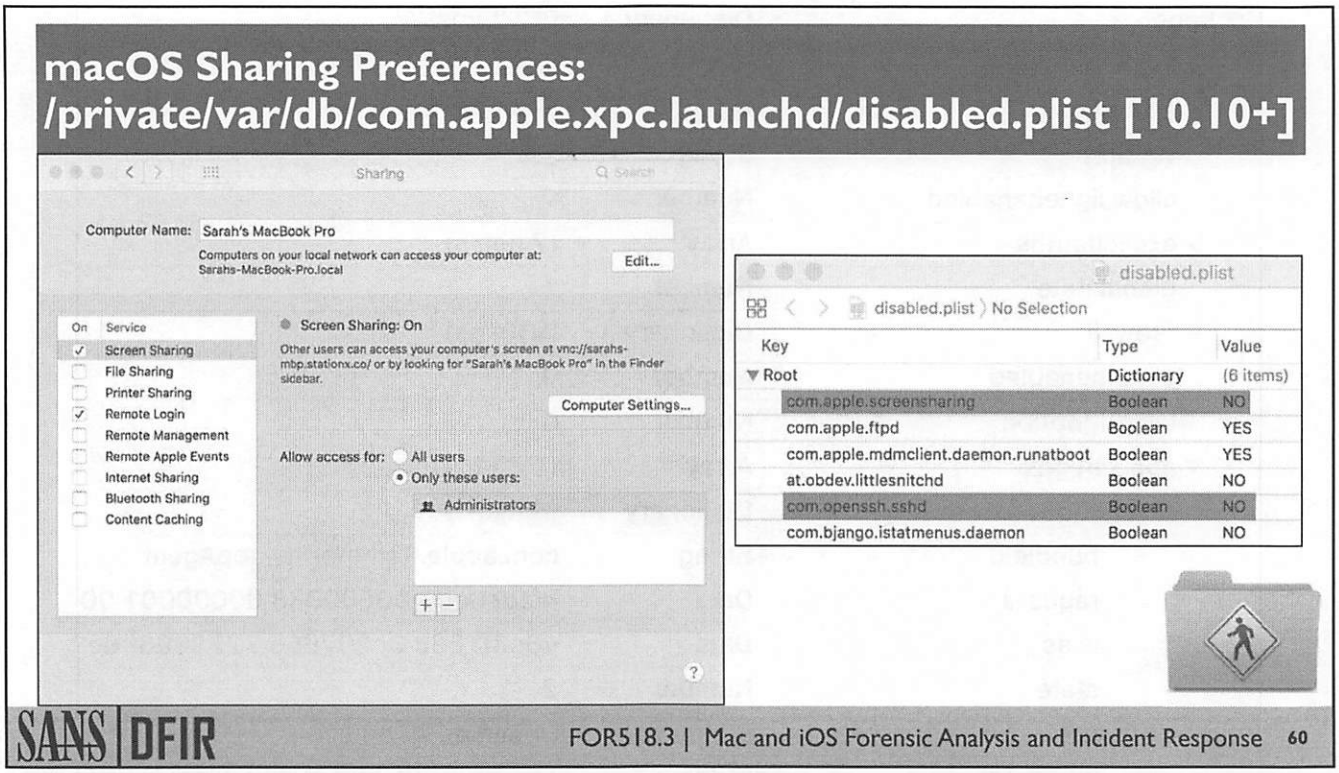
In the screenshot of the application firewall, there are two checkboxes the users can configure. The following keys hold this configuration:

- The `allowsignedenabled` key determines if the checkbox to allow signed software to receive incoming connections is checked (1 = checked).
- The `allowdownloadsignedenabled` key determines if the checkbox to allow downloaded signed software to receive incoming connections is checked (1 = checked).
- The `stealthenabled` key determines if the stealth enabled checkbox was checked (1 = checked). Enabling stealth mode ignores acknowledgement from network utilities like ping.

The `applications` key lists the applications configured in the firewall, as shown in the screenshot on the previous slide. Each application item contains the applications bundle ID, alias data, and the state. A state of "0" allows incoming connections, while a state of "2" blocks incoming connections to the application.

The remote access options can be found in the `Sharing` preferences pane.

▽ Root	Dictionary	(12 items)
loggingenabled	Number	1
▷ exceptions	Array	(9 items)
version	String	1.6
allowsignedenabled	Number	1
▷ explicitauths	Array	(7 items)
globalstate	Number	1
▷ firewall	Dictionary	(9 items)
stealthenabled	Number	0
loggingoption	Number	0
▽ applications	Array	(4 items)
▽ Item 0	Dictionary	(4 items)
bundleid	String	com.apple.RemoteDesktopAgent
reqdata	Data	<fade0c00 00000038 00000001 00
alias	Data	<3c3f786d 6c207665 7273696f 6e3
state	Number	2
▽ Item 1	Dictionary	(4 items)
bundleid	String	com.apple.WebKit.Networking
reqdata	Data	<fade0c00 00000038 00000001 00
alias	Data	<3c3f786d 6c207665 7273696f 6e3
state	Number	0
▽ Item 2	Dictionary	(4 items)
bundleid	String	com.getdropbox.dropbox
reqdata	Data	<fade0c00 000000a4 00000001 00
alias	Data	<3c3f786d 6c207665 7273696f 6e3
state	Number	0
▽ Item 3	Dictionary	(4 items)
bundleid	String	com.blackbagtech.BlackLight
reqdata	Data	<fade0c00 000000a8 00000001 00
alias	Data	<3c3f786d 6c207665 7273696f 6e3
state	Number	0
firewallunload	Number	0
allowdownloadsignedenabled	Number	1



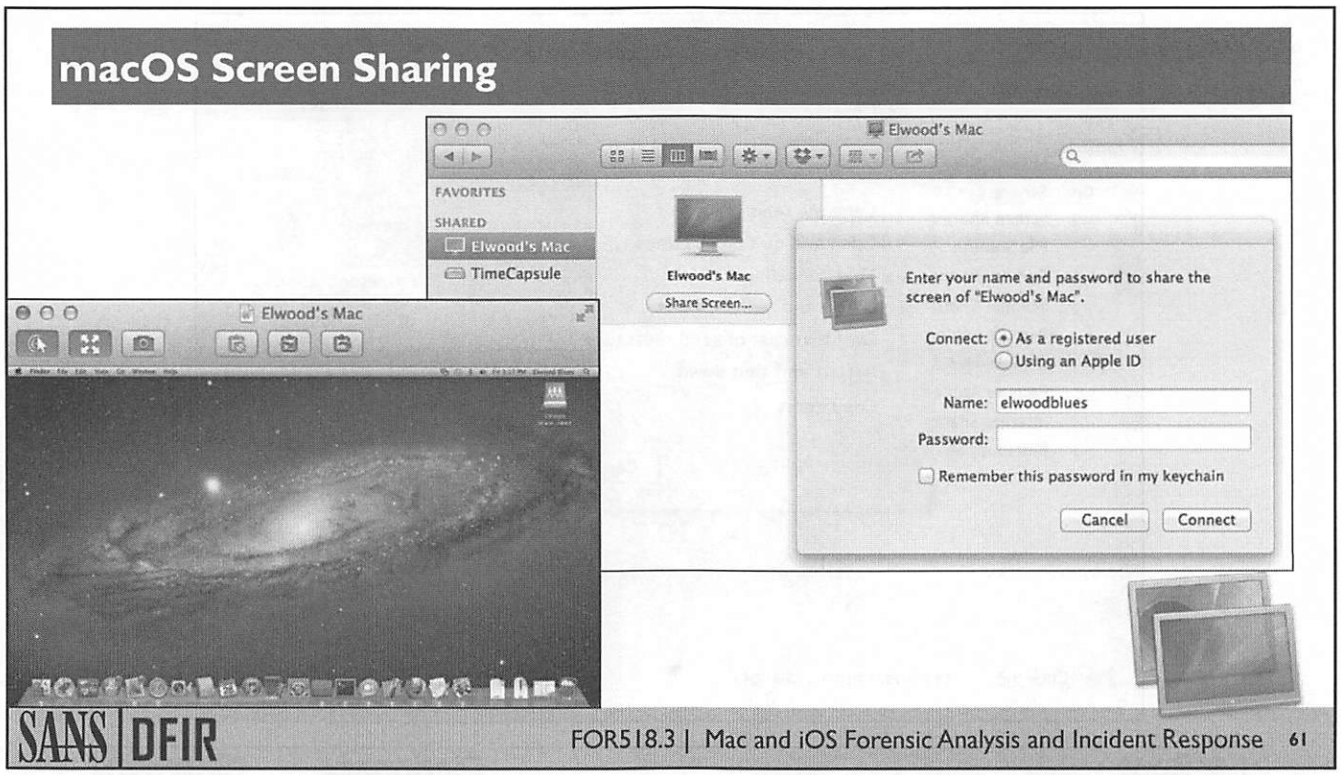
The `Sharing` preferences pane shown in the screenshot above contains the items that can be shared from the system. By default, none of these are enabled.

The `overrides.plist/disabled.plist` file contains many configuration keys for various sharing settings. The `com.apple.screensharing` key can contain a Yes (1) or No (0). If Screen Sharing is enabled, the key will show "No" or 0, meaning it is NOT disabled—therefore enabled.

Remote Login allows a user to remotely log in to the system via the SSH (Secure Shell) protocol.

If the bundle ID for the service does not appear in this list at all, it was likely not checked ever in the past and therefore never enabled.

```
com.openssh.sshd = Remote Login
com.apple.screensharing = Screen Sharing
```



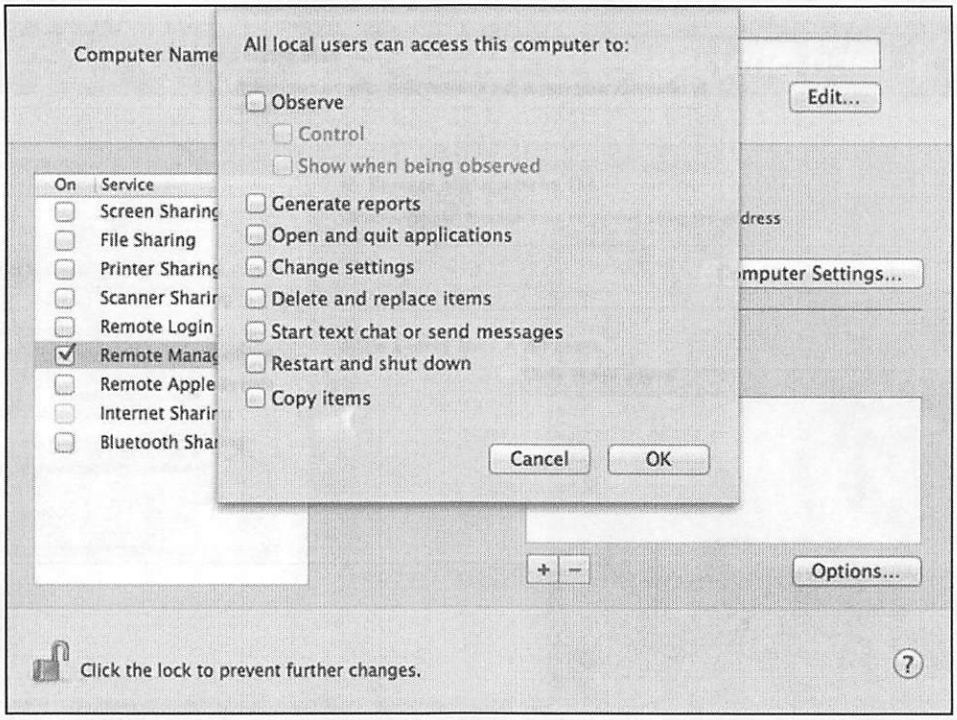
The Screen Sharing application, located in `/System/Library/CoreServices/` rather than `/Applications`, is used to connect to other systems using the VNC protocol.

The `com.apple.RemoteManagement.plist` is created in the `/Library/Preferences/` directory when the Screen Sharing or Remote Management options are selected in the Sharing preferences pane.

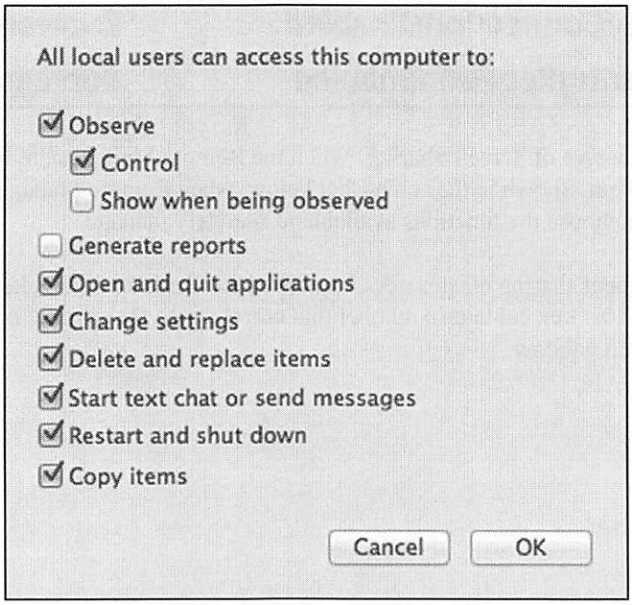
▼Root	Dictionary	(2 items)
VNCLegacyConnectionsEnabled	Boolean	YES
ScreenSharingReqPermEnabled	Boolean	NO

Remote Management is inclusive of Screen Sharing. When the Remote Management box is checked, the Screen Sharing box will be grayed out. Shown in the screenshot below, when Remote Management is checked, a box will pop up allowing the user to choose the functions available to remotely manage.

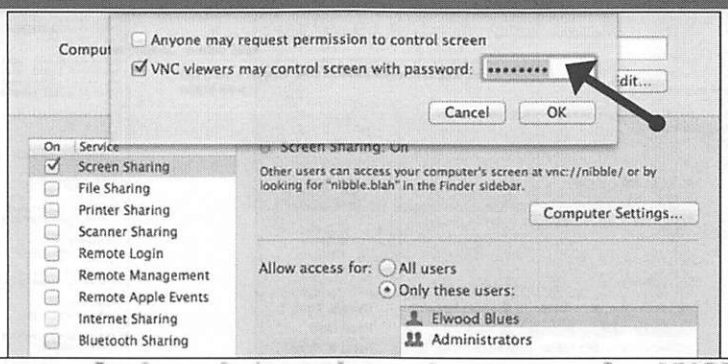
Selecting Remote Management sharing creates new Apple Remote Desktop (ARD) keys. The `ARD_AllLocalUserPrivs` key contains a number that correlates to the specific options selected in the Remote Management Options pop-up window.



Root	Dictionary	(5 items)
ARD_AllLocalUsersPrivs	Number	239
VNCLegacyConnectionsEnabled	Boolean	YES
LoadRemoteManagementMenuExtra	Boolean	NO
ARD_AllLocalUsers	Boolean	YES
ScreenSharingReqPermEnabled	Boolean	NO



Sharing Preferences: Screen Sharing /Library/Preferences/com.apple.VNCSettings.txt

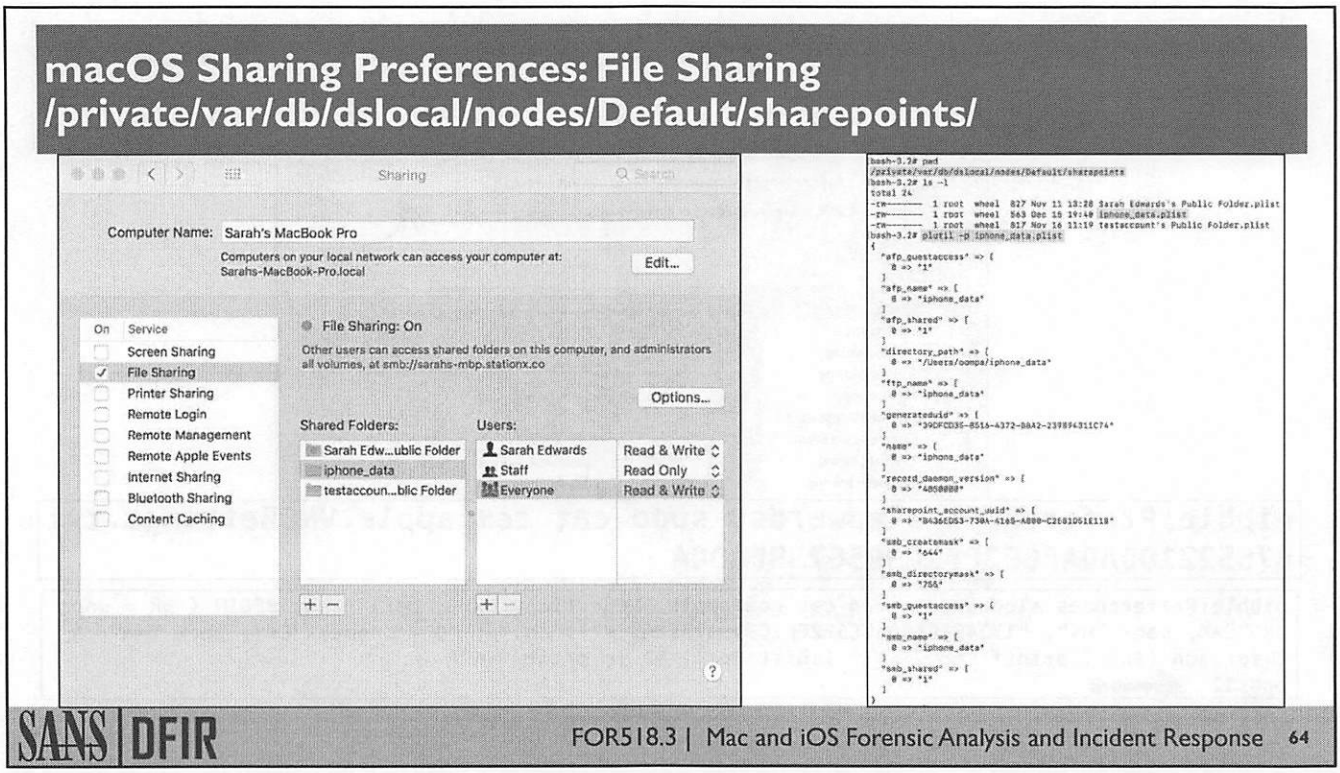



```
nibble:Preferences sledwards$ sudo cat com.apple.VNCSettings.txt
6755221DBA9AF6E2FF1C39567390ADCA
```

```
nibble:Preferences sledwards$ sudo cat com.apple.VNCSettings.txt | perl -wne 'BEGIN { @k = unpa
ck "C*", pack "H*", "1734516E8BA8C5E2FF1C39567390ADCA"}; chomp; @p = unpack "C*", pack "H*", $_
; foreach (@k) { printf "%c", $_ ^ (shift @p || 0) }; print "\n"'
pass123
```


The `com.apple.VNCSettings.txt` text file contains the XOR'ed "encrypted" password to access the system via a VNC viewer client. We can use the Perl script created by Ben Low to "decrypt" the VNC password using the XOR key "`1734516E8BA8C5E2FF1C39567390ADCA`".

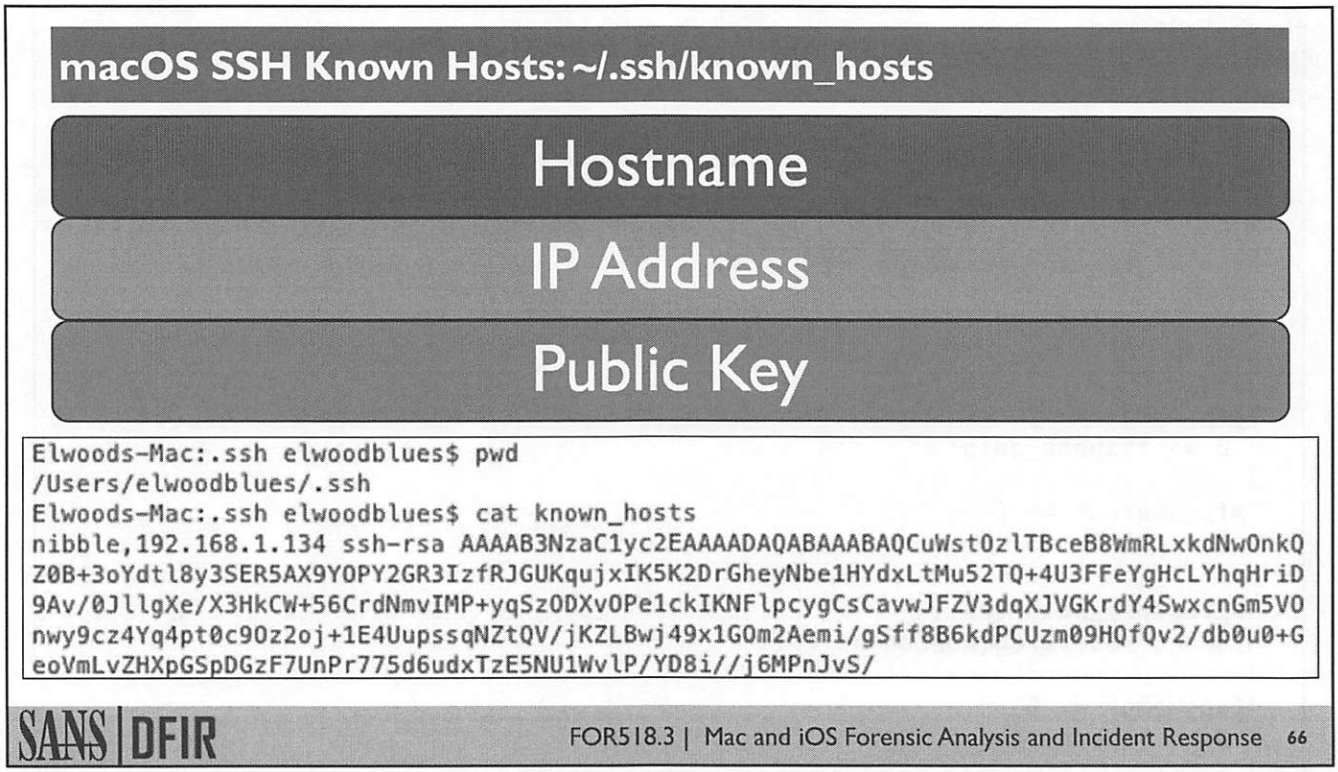
```
cat com.apple.VNCSettings.txt | perl -wne 'BEGIN { @k = unpack "C*", pack
"H*", "1734516E8BA8C5E2FF1C39567390ADCA"}; chomp; @p = unpack "C*", pack
"H*", $_; foreach (@k) { printf "%c", $_ ^ (shift @p || 0) }; print "\n"'
```



File Sharing allows users to share files on the network or with other users on the system. These shared folders and their metadata can be viewed as plist files in the `/private/var/db/dslocal/nodes/Default/sharepoints/` directory. The example above shows that the folder `iphone_data` was shared along with the related permissions and directory path.

Look for com.apple.smbd and/or com.apple.AppleFileServer as the bundle ids for these in overrides.plist or disabled.plist files.

```
[bash-3.2# pwd
 /private/var/db/dslocal/nodes/Default/sharepoints
[bash-3.2# ls -l
total 24
-rw-------  1 root  wheel  827 Nov 11 13:28 Sarah Edwards's Public Folder.plist
-rw-------  1 root  wheel  563 Dec 15 19:40 iphone_data.plist
-rw-------  1 root  wheel  817 Nov 16 11:19 testaccount's Public Folder.plist
[bash-3.2# plutil -p iphone_data.plist
{
  "afp_guestaccess" => [
    0 => "1"
  ]
  "afp_name" => [
    0 => "iphone_data"
  ]
  "afp_shared" => [
    0 => "1"
  ]
  "directory_path" => [
    0 => "/Users/oompa/iphone_data"
  ]
  "ftp_name" => [
    0 => "iphone_data"
  ]
  "generateduid" => [
    0 => "39DFCD35-8516-4372-B8A2-239894311C74"
  ]
  "name" => [
    0 => "iphone_data"
  ]
  "record_daemon_version" => [
    0 => "4850000"
  ]
  "sharepoint_account_uuid" => [
    0 => "B436ED53-730A-4165-A800-C2E81D51E11B"
  ]
  "smb_createmask" => [
    0 => "644"
  ]
  "smb_directorymask" => [
    0 => "755"
  ]
  "smb_guestaccess" => [
    0 => "1"
  ]
  "smb_name" => [
    0 => "iphone_data"
  ]
  "smb_shared" => [
    0 => "1"
  ]
}
```



If a user is more computer savvy, they may use SSH to connect to other systems. The SSH `known_hosts` file contains a hostname and/or IP address and a public key. This file is not a definitive way to show all systems that a user connected to, but will contain those that have a saved public key.

By default, the hostnames and IP address should be human readable. If the user has configured their `/etc/ssh/ssh_config` file to set `HashKnownHosts` to yes, this data will be hashed.

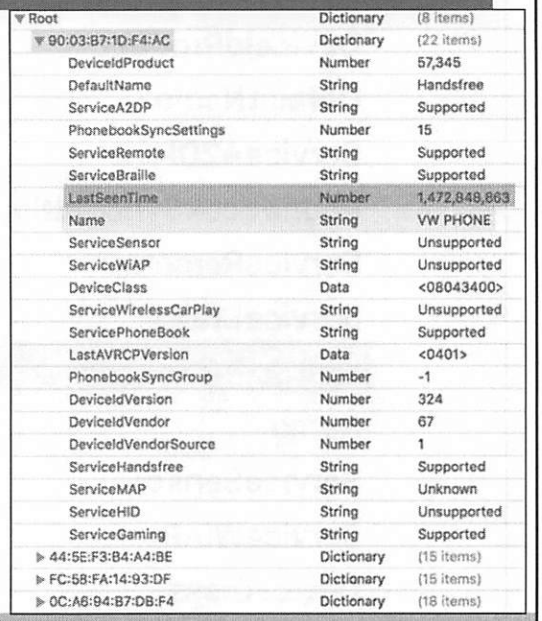
iOS Bluetooth Devices

Location:

- Look for directories/files named:
  - *MobileBluetooth*
  - *com.apple.Bluetooth*
  - *Bluetooth*

Paired Devices

- Cars
- iDevices
- Computers
- Speakers/Headphones
- Etc.



Root	Dictionary	(8 items)
90:03:B7:1D:F4:AC	Dictionary	(22 items)
DeviceIdProduct	Number	57,345
DefaultName	String	Handsfree
ServiceA2DP	String	Supported
PhonebookSyncSettings	Number	15
ServiceRemote	String	Supported
ServiceBraille	String	Supported
LastSeenTime	Number	1,472,848,863
Name	String	VW PHONE
ServiceSensor	String	Unsupported
ServiceWiAP	String	Unsupported
DeviceClass	Data	<08043400>
ServiceWirelessCarPlay	String	Unsupported
ServicePhoneBook	String	Supported
LastAVRCPVersion	Data	<0401>
PhonebookSyncGroup	Number	-1
DeviceIdVersion	Number	324
DeviceIdVendor	Number	67
DeviceIdVendorSource	Number	1
ServiceHandsfree	String	Supported
ServiceMAP	String	Unknown
ServiceHID	String	Unsupported
ServiceGaming	String	Supported
44:5E:F3:B4:A4:BE	Dictionary	(15 items)
FC:58:FA:14:93:DF	Dictionary	(15 items)
0C:A6:94:B7:DB:F4	Dictionary	(18 items)

Bluetooth information can be used to determine if this device has been connected to anything of investigative interest, such as another computer, device, or a vehicle.

The data is stored in various places, as noted in the slide, but contains generally the same information. Data found in the `com.apple.MobileBluetooth.ledevices.plist` or `com.apple.MobileBluetooth.ledevices.paired.db` contains information for paired Bluetooth devices, while data found in the `com.apple.MobileBluetooth.device.plist` and `com.apple.MobileBluetooth.ledevice.other.db` contain information about devices that are just near this device.

In the screenshot above is an example of where this device connected to a VW car. The Volkswagen cars tend to use the generic "`VW PHONE`" as their device name.

▼ Root	Dictionary	(8 items)
▼ 90:03:B7:1D:F4:AC	Dictionary	(22 items)
DeviceIdProduct	Number	57,345
DefaultName	String	Handsfree
ServiceA2DP	String	Supported
PhonebookSyncSettings	Number	15
ServiceRemote	String	Supported
ServiceBraille	String	Supported
LastSeenTime	Number	1,472,848,863
Name	String	VW PHONE
ServiceSensor	String	Unsupported
ServiceWiAP	String	Unsupported
DeviceClass	Data	<08043400>
ServiceWirelessCarPlay	String	Unsupported
ServicePhoneBook	String	Supported
LastAVRCPVersion	Data	<0401>
PhonebookSyncGroup	Number	-1
DeviceIdVersion	Number	324
DeviceIdVendor	Number	67
DeviceIdVendorSource	Number	1
ServiceHandsfree	String	Supported
ServiceMAP	String	Unknown
ServiceHID	String	Unsupported
ServiceGaming	String	Supported
▶ 44:5E:F3:B4:A4:BE	Dictionary	(15 items)
▶ FC:58:FA:14:93:DF	Dictionary	(15 items)
▶ 0C:A6:94:B7:DB:F4	Dictionary	(18 items)

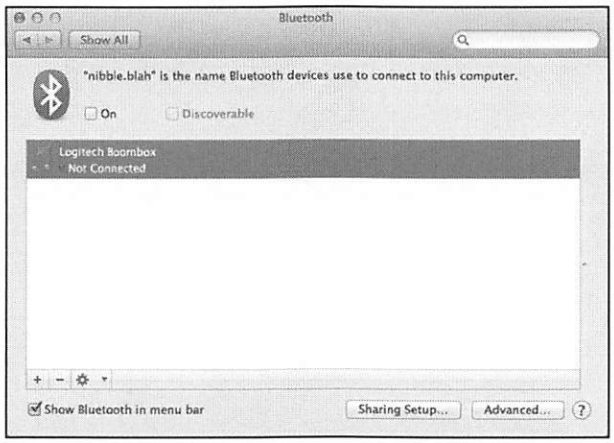
macOS Bluetooth: /Library/Preferences/com.apple.Bluetooth.plist

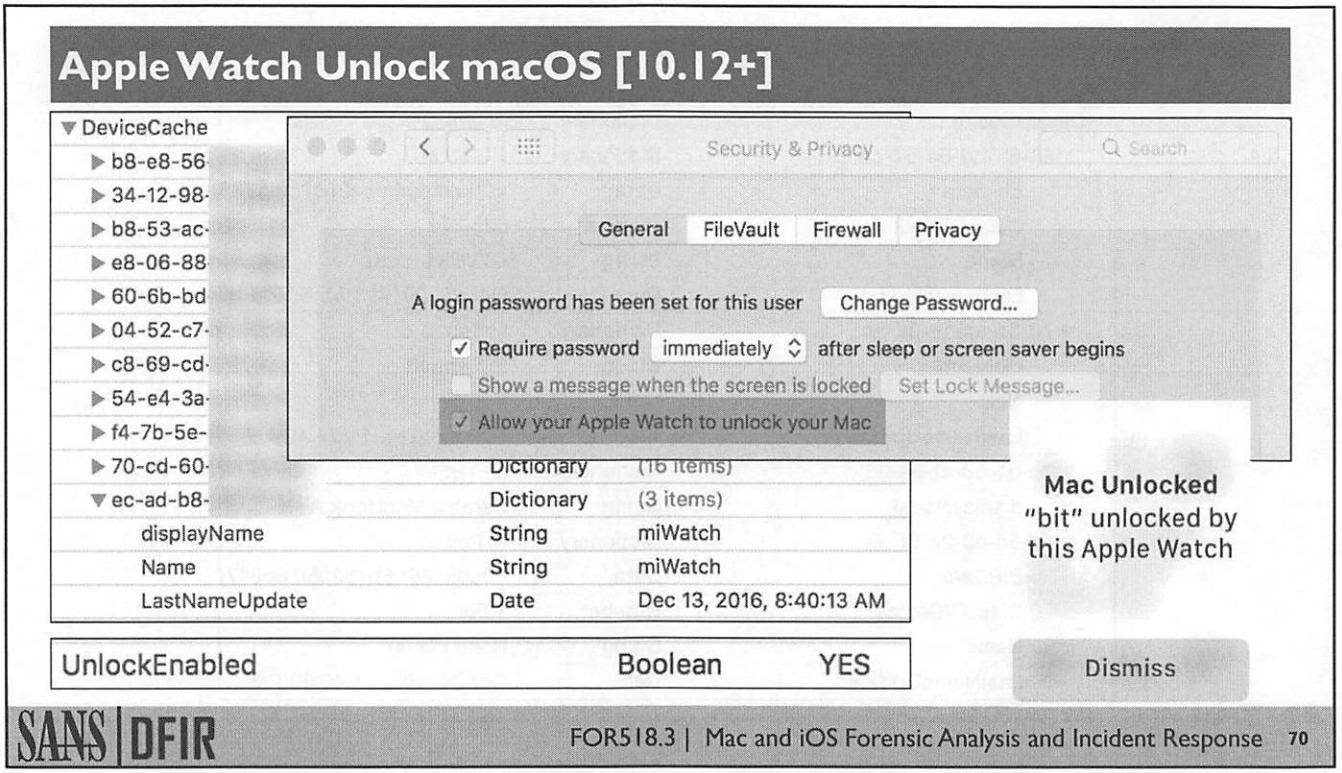
▼ DeviceCache	Dictionary	(5 items)
▼ 60-6b-bd-0e-57-c8	Dictionary	(4 items)
EIRData	Data	<0d094454 56426c75 65746
ClassOfDevice	Number	525,372
Name	String	DTVBluetooth
LastNameUpdate	Date	Oct 26, 2014, 1:55:55 PM
▼ 70-3e-ac-16-05-bc	Dictionary	(3 items)
displayName	String	miPhone6
Name	String	iPhone
LastNameUpdate	Date	Nov 5, 2014, 11:24:42 AM
▼ 7c-d1-c3-df-64-68	Dictionary	(1 item)
displayName	String	Sarah's MacBook Air
▼ cc-6d-a0-2c-96-2e	Dictionary	(4 items)
EIRData	Data	<0c09526f 6b752050 6c617
ClassOfDevice	Number	1,060
Name	String	Roku Player
LastNameUpdate	Date	Oct 26, 2014, 1:49:30 PM

Configured Bluetooth devices may be shown and configured in the window presented in the Bluetooth preference pane (shown below). The `com.apple.bluetooth.plist` property list located in the `/Library/Preferences/` directory contains the Bluetooth device cache in the `DeviceCache` key and the paired devices in the `PairedDevices` key. This `DeviceCache` key contains a history of the Bluetooth devices connected to the system and the `ClassOfDevice` associated with it. The device class (`ClassOfDevice` key) is the type of device that was paired, such as headphones or a mouse. The device class can be reversed using the Bluetooth header files located in the `/System/Library/Frameworks/IOBluetooth.framework/Headers/` directory.

- `Bluetooth.h`
- `BluetoothAssignedNumbers.h`.

10.8 introduces more information in the `com.apple.Bluetooth.plist`. Each Bluetooth MAC address under the `DeviceCache` key may contain detailed information about the device, such as the device time and last communication timestamps. This is a good place to look for other devices a user may have had near them.

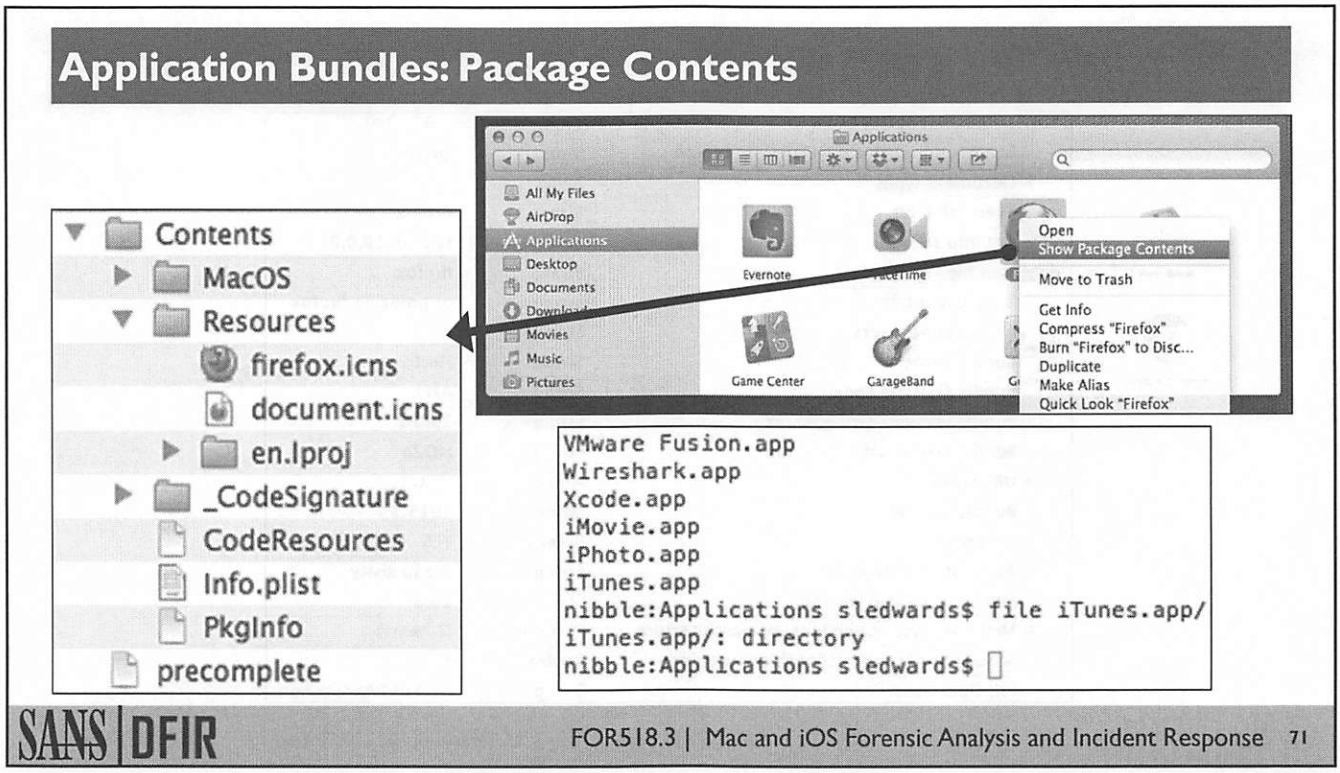




In 10.12, a new unlock capability was introduced to allow Mac computers to be unlocked using the user's Apple Watch. This can be seen in the Bluetooth settings on the macOS settings in the `com.apple.Bluetooth.plist` file. If the key "`UnlockEnabled`" is set to yes, it is enabled.

To unlock the Mac, Bluetooth and Wi-Fi needs to be on and two-factor authentication enabled for the Apple ID used.

Reference:
`https://support.apple.com/en-us/HT206995`



Applications on OS X systems come in a packaged format that appears as one file to the user in the Finder window. The top screenshot shows some of the applications available in the `/Applications` directory, each being accessible by double-clicking the application icon. Each one of these icons is actually the application bundle. It is worth noting that applications are not required to be run from the `Applications` directory. The lower screenshot shows what these applications look like in the Terminal window. Each application has the `.app` file extension and is a directory containing other files.

Application bundles are directories that contain everything an application needs to execute, such as:

- Executable Code
- Resource Files
- Support Files

These files can be accessed via the Finder application by "right-clicking" (Control+clicking) the "`Show Package Contents`", as shown in the screenshot above. Each application bundle contains the same basic directory and file structure contained in the "`Contents`" directory.

- `Info.plist` File: Information Property List
- `MacOS` Directory: Contains executable code
- `Resources` Directory: Contains resource files

References:

Bundle Programming Guide: Apple Developer Documentation
`http://developer.apple.com/library/mac/#documentation/CoreFoundation/Conceptual/CFBundles/AboutBundles/AboutBundles.html`

Application Bundles: Apple Developer Documentation
`http://developer.apple.com/library/mac/#documentation/CoreFoundation/Conceptual/CFBundles/BundleTypes/BundleTypes.html#//apple_ref/doc/uid/10000123i-CH101-SW13`

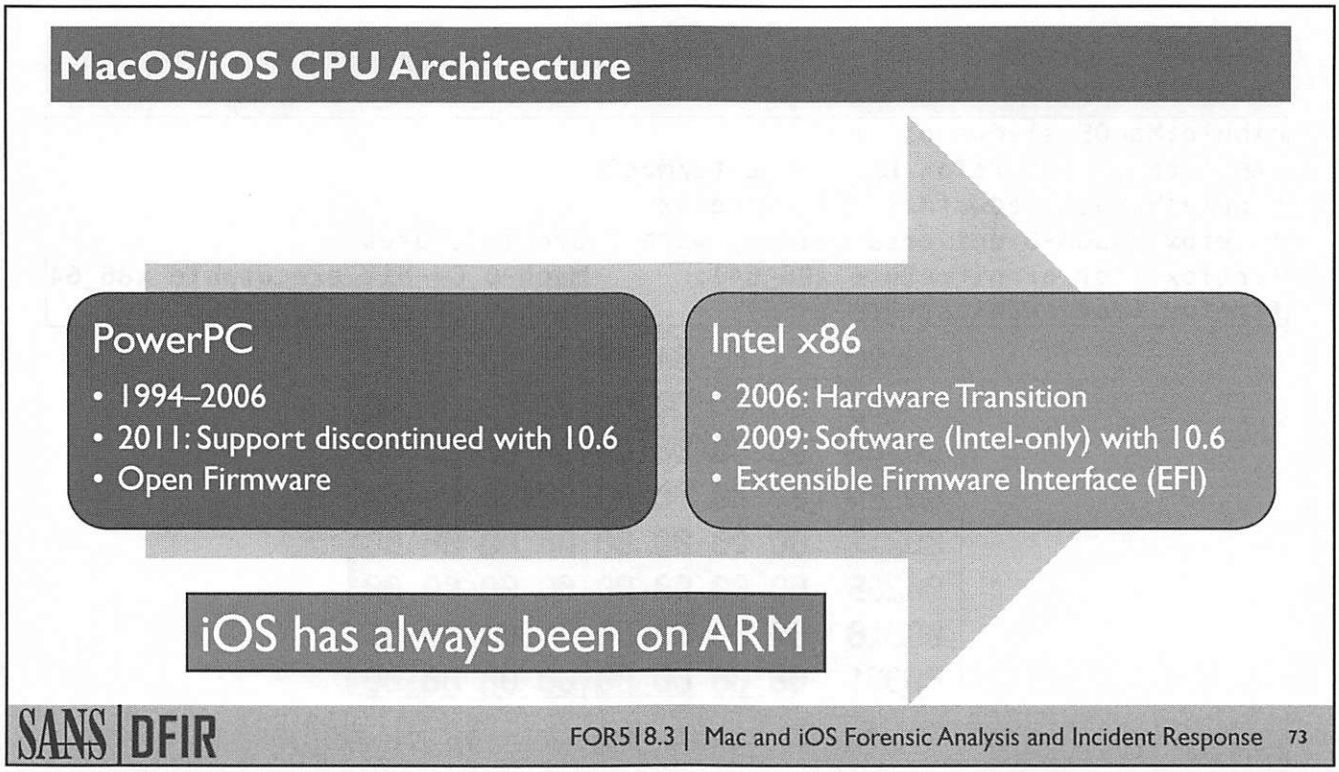
Application Bundles: Information Property List—Info.plist

Information Property List	Dictionary	(19 items)
Localization native development region	String	English
▶ Document types	Array	(7 items)
Executable file	String	firefox
Get Info string	String	Firefox 19.0.2
Icon file	String	firefox
Bundle identifier	String	org.mozilla.firefox
InfoDictionary version	String	6.0
Bundle name	String	Firefox
Bundle OS Type code	String	APPL
Bundle versions string, short	String	19.0.2
Bundle creator OS Type code	String	MOZB
▶ URL types	Array	(4 items)
Bundle version	String	1913.3.7
Scriptable	Boolean	YES
Application Category	String	Productivity
Minimum system version	String	10.6
▶ Minimum system versions, per-architecture	Dictionary	(2 items)
NSSupportsAutomaticGraphicsSwitching	Boolean	YES
Principal class	String	GeckoNSApplication

The Information Property List (`Info.plist`) contains XML formatted data that describes the application.

- Bundle Name
- Bundle Version
- Bundle Identifier (Reverse DNS format)
- Executable Filename

The example shows an `Info.plist` file for the Firefox application, version 19.0.2. Other items in the file may be optional but still useful to an investigator to determine the purpose of a specific application. The "`Application Category`" is how the developer determines what type of application it is, while the "`Document types`" and "`URL types`" determine what files or URLs this program supports. The application may also state what versions of OS X it will run on, listing minimum and system versions and architecture types.



After the introduction of Unix core components to the operating system, Apple Computer made the transition from PowerPC to Intel x86 architecture.

PowerPC (Performance Optimization With Enhanced RISC – Performance Computing) was used from 1994 to 2006. PowerPC was no longer supported with OS X Snow Leopard (10.6). This architecture used the Open Firmware interface, similar to a PC's BIOS. The last version of OS X that was PowerPC-only was OS X Panther (10.3). This interface was accessible using the CMD+Option+O+F.

The Intel x86 transition started in 2006. OS X Tiger (10.4) was the first available OS X version that one could use on x86 architecture. Intel-based Macs use the Extensible Firmware Interface (EFI), also similar to a PC's BIOS.

Open Firmware and EFI communicate between the hardware and the operating system to provide drivers and a pre-OS environment. EFI has additional advantages, such as the ability to boot from USB and larger disks.

While macOS has changed over time, iOS was always on ARM architecture, either 32- or 64-bit, depending on the age of the device.

Mach-O Executables

```
nibble:MacOS sledwards$ pwd
/Applications/Firefox.app/Contents/MacOS
nibble:MacOS sledwards$ file firefox
firefox: Mach-O universal binary with 2 architectures
firefox (for architecture x86_64):      Mach-O 64-bit executable x86_64
firefox (for architecture i386):        Mach-O executable i386
```

```
00000  CA FE BA BE 00 00 00 02
00053  00 00 00 00 00 00 00 00
00106  00 00 00 00 00 00 00 00
00159  00 00 00 00 00 00 00 00
00212  00 00 00 00 00 00 00 00
00265  00 00 00 00 00 00 00 00
00318  00 00 00 00 00 00 00 00
00371  00 00 00 00 00 00 00 00
```


Mach-O (Mach Object) executable files are the main type of binary used for Mac applications. Mach-O binaries can support multiple architectures. The top screenshot shows the file command used on the Firefox executable. These are called universal binaries, or "fat" binaries.

- x86_64 executables can be used on Macs with the 64-bit Intel processor.
  - It is worth noting that some 64-bit executables can run on 32-bit kernels—visit the AppleInsider.com article listed in the References section for more information.
- i386 executables can be used on Macs with the 32- and 64-bit Intel processors.
- iOS devices may run 32-bit or 64-bit ARM executables, as shown here for Mobile Safari.

```
bit:temp oompa$ file MobileSafari
MobileSafari: Mach-O 64-bit executable arm64
```

Fat binaries include multiple architectures, such as PowerPC and Intel (these are sometimes called Universal binaries), or Intel 32-bit and 64-bit, as shown in the screenshot above.

The Mach-O binary uses many signatures depending on the architecture:

- `0xCAFEBABE`: Fat binary
- `0xFEEDFACE`: 32-bit
- `0xFEEDFACF`: 64-bit
- `0xCEFAEDFE`: 32-bit, little endian
- `0xCFFAEDFE` : 64-bit, little endian

References:
OS X Mach-O File Format Reference: Apple Developer Documentation
`https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/MachOTopics/0-Introduction/introduction.html`
`http://appleinsider.com/articles/08/10/28/road_to_mac_os_x_snow_leopard_64_bit_to_the_kernel.html`

Kernel Extensions: /System/Library/Extensions/*.kext

- Dynamically loaded executable code in kernel space
  - /private/var/db/loadedkextmt.plist
  - "kextstat" Command – Live Response (example below)
  - Low-Level Device Drivers, Network Filters, File Systems… Keyloggers? … Malware?
- 10.13 – Introduced User-Approved Extension Loading

```
159    0 0xffffff7f845f7000 0x5000     0x5000     com.apple.Dont_Steal_Mac_OS_X (7.0.0) EBA46A07-5CA9-3449-AB3C-194D7EDBDEC6 <14 9 8 5 3 1>
160    1 0xffffff7f845fc000 0x5000     0x5000     com.objective-see.lulu (1.1.2) 55DEAB86-7E59-3F37-A64E-0CE0D0C5AB51 <8 6 5 3 1>
161    0 0xffffff7f84601000 0x18000    0x18000    com.apple.fileutil (18.306.12) C5EF45E1-B31A-3DC0-827C-5057DAAADDE6 <6 5 3 2 1>
162    0 0xffffff7f84619000 0x12000    0x12000    com.apple.driver.pmtelemetry (1) 90FED9E0-C8C8-36C4-8231-99FF16B46CB0 <8 7 6 5 3>
163    0 0xffffff7f8462b000 0xa000     0xa000     com.apple.iokit.IOBluetoothSerialManager (6.0.9f2) 6D248369-38DB-3948-B7B6-C62D339B55A7 <85 8 6 5 3 1>
164    0 0xffffff7f84635000 0xb000     0xb000     com.apple.driver.AppleSSE (1.0) 820E89F2-2F5E-3673-AF12-FAB3726810D2 <29 28 27 8 6 5 3 1>
165    0 0xffffff7f84640000 0x6000     0x6000     com.apple.iokit.IOUserEthernet (1.0.1) 67F54A00-494C-357F-B4FF-6A291C66C4DA <23 7 6 5 3 1>
166    0 0xffffff7f84646000 0x9000     0x9000     com.apple.driver.AppleHV (1) 70BE21E9-ED1C-3847-841B-F3692643F387 <8 7 6 5 3 1>
167    0 0xffffff7f8464f000 0x17f000   0x17f000   at.obdev.nke.LittleSnitch (5210) 0BEBCAD5-9FC2-35B4-A87D-08B14BCB15F5 <8 6 5 3 1>
168    2 0xffffff7f816f8000 0x7000     0x7000     com.apple.iokit.IOEthernetAVBController (1.1.0) 43E4FF0D-16A8-32EB-AAC0-73ADB153914E <23 6 5 3 1>
169    1 0xffffff7f847ce000 0x9b000    0x9b000    com.apple.plugin.IOgPTPPlugin (700.7) F2799BC4-8FEA-392F-B132-F6A67CB4BC1B <168 44 24 23 11 7 6 5 3 1>
170    0 0xffffff7f84869000 0xf000     0xf000     com.apple.iokit.IOAVBFamily (710.1) 2B2C886C-C656-3782-998D-79172349C318 <169 168 24 23 7 6 5 3 1>
171    0 0xffffff7f84878000 0x3000     0x3000     com.apple.AGDCPluginDisplayMetrics (3.28.4) 80FF8D30-0F1F-3EAD-8C06-126254E42C02 <122 8 7 6 5 3 1>
172    1 0xffffff7f8487b000 0x3000     0x3000     com.apple.driver.AppleGraphicsControl (3.28.4) 200F1816-A734-3977-B7AE-7A6B2363059F <8 6 5 3 1>
173    0 0xffffff7f8487e000 0x12000    0x12000    com.apple.driver.AppleGraphicsDevicePolicy (3.28.4) B966AFB2-2D6A-3A5C-9895-C9A1CC39AB88 <172 122 110 13 12 8 7 6 5 3 1>
174    0 0xffffff7f84890000 0x65000    0x65000    com.apple.filesystems.smbfs (3.3.1) E8502198-6395-35AA-9DBD-35AD93251D37 <108 9 8 7 6 5 3 1>
175    0 0xffffff7f848f5000 0x6000     0x6000     com.getdropbox.dropbox.kext (1.9.1) C83721F1-2B15-3E67-B7C2-D5B0ABA4B08C <8 6 5 2 1>
198    0 0xffffff7f84359000 0x19000    0x19000    com.github.osxfuse.filesystems.osxfuse (3.8.2) 7AA38E46-2C0C-39C5-B790-6451370874DA <8 6 5 3 1>
```

Kernel extensions are similar to loadable kernel modules on Linux systems. They are often used as device drivers, network filters, or support for various file systems. They can also be used maliciously as keyloggers, as shown in the `kextstat` output in the screenshot above. The `com.fsb.kext.logKext` is the Kernel Extension for the open-source keylogger LogKext.

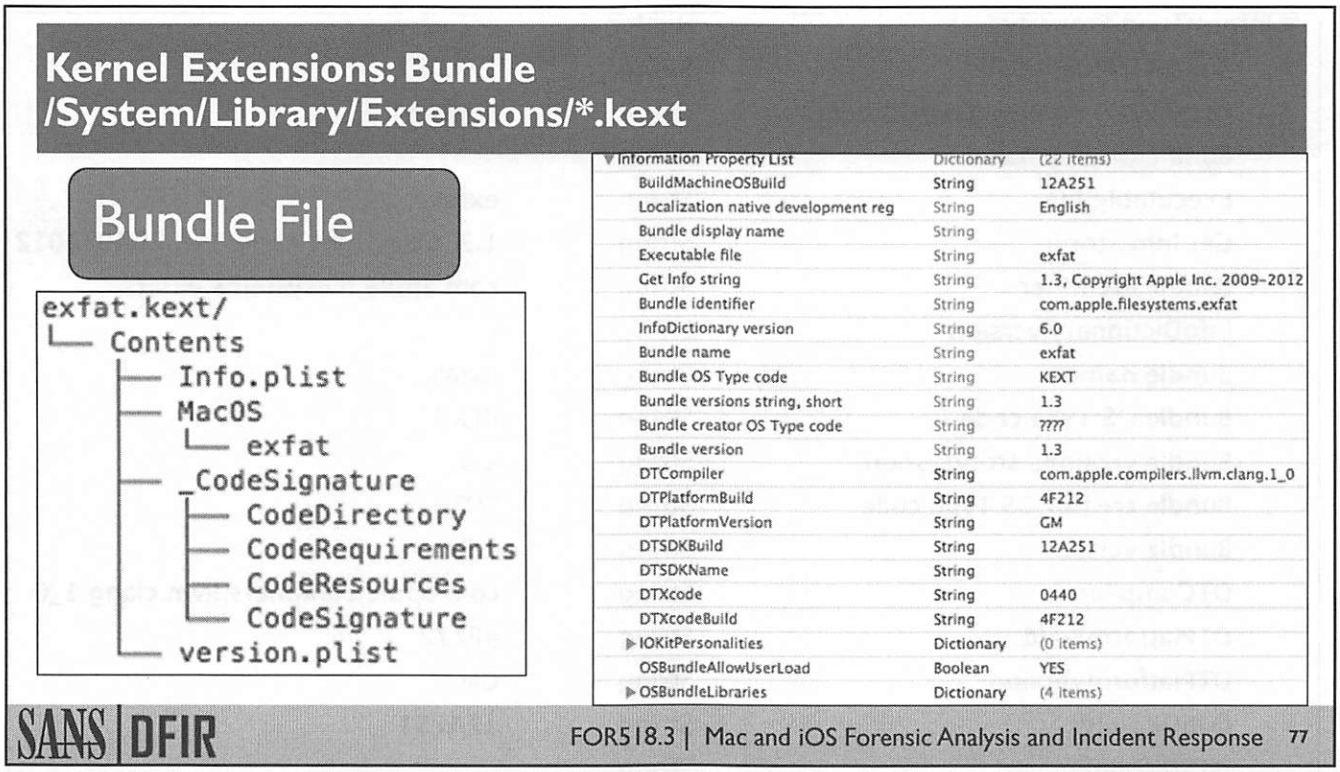
You can use the `kextstat` command on a live system to view the status of the kernel extensions. To see what was loaded on a dead system check out /private/var/db/loadedkextmt.plist.

An older example with an open-source keylogger installed on a system (LogKext):

```
76    0 0xffffff7f81340000 0xa000     0xa000     com.apple.driver.AppleMCCSControl (1.0.24) <55 9 7 5 4 3 1>
77    0 0xffffff7f81214000 0x5000     0x5000     com.apple.driver.AppleUpstreamUserClient (3.5.9) <55 9 8 7 5 4 3 1>
78    1 0xffffff7f813e5000 0xa4000    0xa4000    com.apple.driver.DspFuncLib (2.1.1f12) <67 66 5 4 3 1>
79    0 0xffffff7f81489000 0xaf000    0xaf000    com.apple.driver.AppleHDA (2.1.1f12) <78 67 65 64 57 55 6 5 4 3 1>
81    1 0xffffff7f80f67000 0x5000     0x5000     com.apple.kext.triggers (1.0) <7 6 5 4 3 1>
82    0 0xffffff7f80f6c000 0x9000     0x9000     com.apple.filesystems.autofs (3.0) <81 7 6 5 4 3 1>
83    0 0xffffff7f81631000 0x5000     0x5000     com.vmware.kext.vmmemctl (0068.29.96) <7 5 4 3 1>
85    0 0xffffff7f81637000 0xa000     0xa000     com.vmware.kext.vmhgfs (0068.29.96) <5 4 3 1>
88    0 0xffffff7f80802000 0x4000     0x4000     com.fsb.kext.logKext (2.3) <25 4 3>
```

Reference:
User-Approved Extensions:
`https://developer.apple.com/library/archive/technotes/tn2459/_index.html`

```
159    0 0xffffff7f845f7000 0x5000     0x5000     com.apple.Dont_Steal_Mac_OS_X (7.0.0) E8A46A07-5CA9-3449-AB3C-194D7EDBDEC6 <14 9 8 5 3 1>
160    1 0xffffff7f845fc000 0x5000     0x5000     com.objective-see.lulu (1.1.2) 55DEAB86-7E59-3F37-A64E-0CE0D0C5AB51 <8 6 5 3 1>
161    0 0xffffff7f84601000 0x18000    0x18000    com.apple.fileutil (18.306.12) C5EF45E1-B31A-3DC0-827C-5057DAAADDE6 <6 5 3 2 1>
162    0 0xffffff7f84619000 0x12000    0x12000    com.apple.driver.pmtelemetry (1) 90FED9E0-C0C8-36C4-8231-99FF16B46CB0 <8 7 6 5 3>
163    0 0xffffff7f8462b000 0xa000     0xa000     com.apple.iokit.IOBluetoothSerialManager (6.0.9f2) 6D248369-38DB-3948-B7B6-C62D339B55A7 <85 8 6 5 3 1>
164    0 0xffffff7f84635000 0xb000     0xb000     com.apple.driver.AppleSSE (1.0) 820E89F2-2F5E-3673-AF12-FAB3726810D2 <29 28 27 8 6 5 3 1>
165    0 0xffffff7f84640000 0x6000     0x6000     com.apple.iokit.IOUserEthernet (1.0.1) 67F54A00-494C-357F-B4FF-6A291C66C4DA <23 7 6 5 3 1>
166    0 0xffffff7f84646000 0x9000     0x9000     com.apple.driver.AppleHV (1) 708E21E9-ED1C-3847-841B-F3692643F387 <8 7 6 5 3 1>
167    0 0xffffff7f8464f000 0x17f000   0x17f000   at.obdev.nke.LittleSnitch (5210) 0BEBCAD5-9FC2-35B4-A87D-08B14BCB15F5 <8 6 5 3 1>
168    2 0xffffff7f816f8000 0x7000     0x7000     com.apple.iokit.IOEthernetAVBController (1.1.0) 43E4FF0D-16A8-32EB-AAC0-73ADB153914E <23 6 5 3 1>
169    1 0xffffff7f847ce000 0x9b000    0x9b000    com.apple.plugin.IOgPTPPlugin (700.7) F27998C4-8FEA-392F-B132-F6A67CB4BC1B <168 44 24 23 11 7 6 5 3 1>
170    0 0xffffff7f84869000 0xf000     0xf000     com.apple.iokit.IOAVBFamily (710.1) 2B2C886C-C656-3782-998D-79172349C318 <169 168 24 23 7 6 5 3 1>
171    0 0xffffff7f84878000 0x3000     0x3000     com.apple.AGDCPluginDisplayMetrics (3.28.4) 80FF8D30-0F1F-3EAD-8C06-126254E42C02 <122 8 7 6 5 3 1>
172    1 0xffffff7f8487b000 0x3000     0x3000     com.apple.driver.AppleGraphicsControl (3.28.4) 200F1816-A734-3977-B7AE-7A6B2363059F <8 6 5 3 1>
173    0 0xffffff7f8487e000 0x12000    0x12000    com.apple.driver.AppleGraphicsDevicePolicy (3.28.4) B966AFB2-2D6A-3A5C-9895-C9A1CC39AB88 <172 122 110 13 12 8 7 6 5 3 1>
174    0 0xffffff7f84890000 0x65000    0x65000    com.apple.filesystems.smbfs (3.3.1) E8502198-6395-35AA-9DBD-35AD93251D37 <108 9 8 7 6 5 3 1>
175    0 0xffffff7f848f5000 0x6000     0x6000     com.getdropbox.dropbox.kext (1.9.1) C83721F1-2B15-3E67-B7C2-D5B0ABA4B08C <8 6 5 2 1>
198    0 0xffffff7f84359000 0x19000    0x19000    com.github.osxfuse.filesystems.osxfuse (3.8.2) 7AA38E46-2C0C-39C5-B790-6451370874DA <8 6 5 3 1>
```

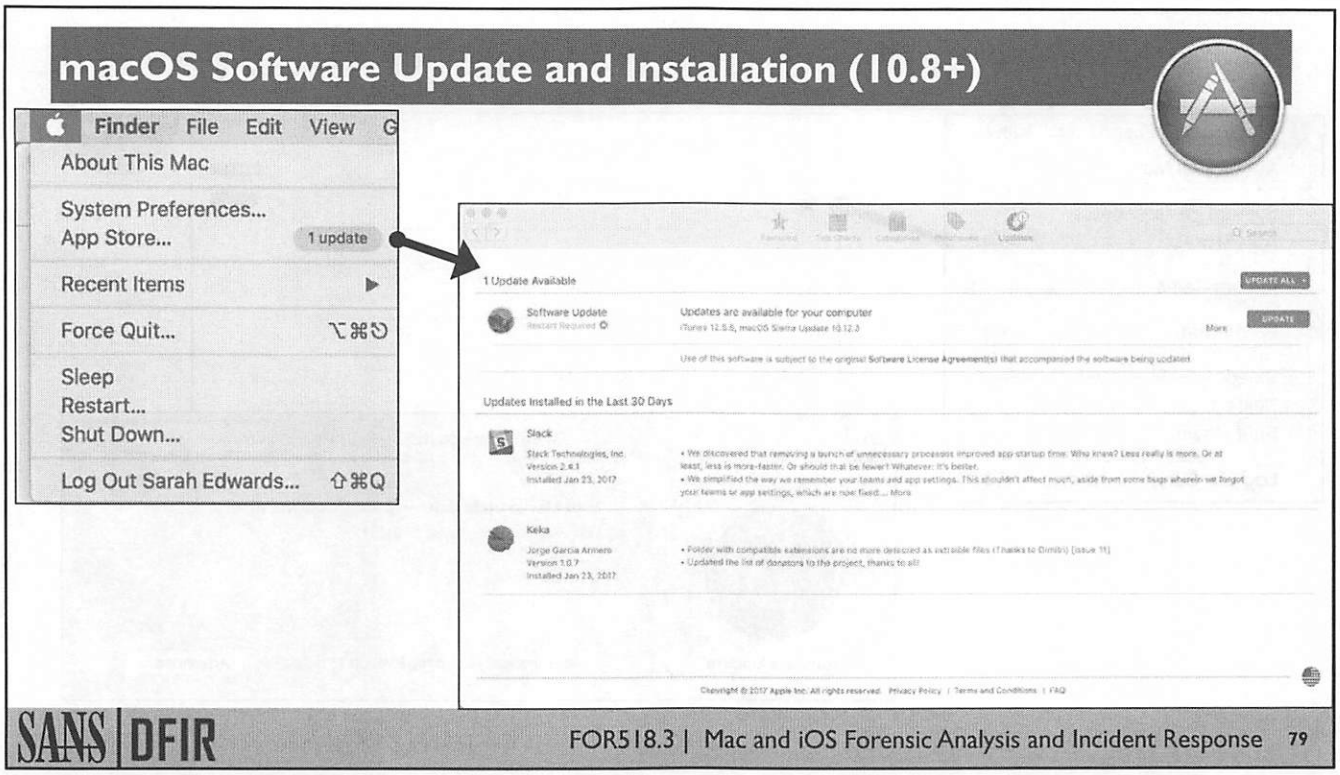


Each kernel extension is a bundle file, meaning they appear as a single file in the Finder application.

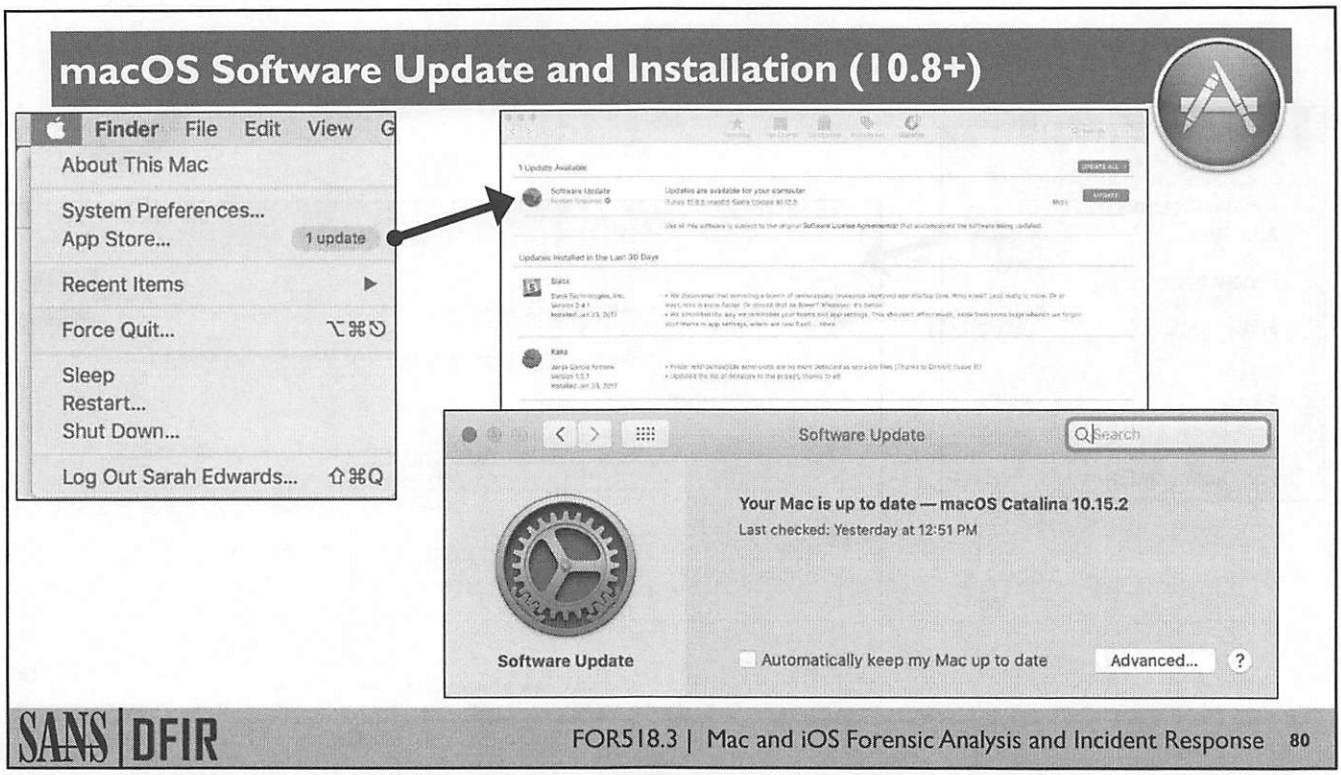
The screenshot on the left shows the file structure of the `exfat` file system kernel extension. Each will contain a Contents directory with an `Info.plist` file, as shown on the right. The `Info.plist` file contains the identifying information of the kernel extension, including the version number, executable filename, and build information.

Reference:
`https://developer.apple.com/library/content/documentation/Darwin/Conceptual/KernelProgramming/Extend/Extend.html`

Information Property List	Dictionary	(22 items)
BuildMachineOSBuild	String	12A251
Localization native development reg	String	English
Bundle display name	String	
Executable file	String	exfat
Get Info string	String	1.3, Copyright Apple Inc. 2009-2012
Bundle identifier	String	com.apple.filesystems.exfat
InfoDictionary version	String	6.0
Bundle name	String	exfat
Bundle OS Type code	String	KEXT
Bundle versions string, short	String	1.3
Bundle creator OS Type code	String	????
Bundle version	String	1.3
DTCompiler	String	com.apple.compilers.llvm.clang.1_0
DTPlatformBuild	String	4F212
DTPlatformVersion	String	GM
DTSDKBuild	String	12A251
DTSDKName	String	
DTXcode	String	0440
DTXcodeBuild	String	4F212
▶ IOKitPersonalities	Dictionary	(0 items)
OSBundleAllowUserLoad	Boolean	YES
▶ OSBundleLibraries	Dictionary	(4 items)



The Apple menu contains the item "`Software Update...`". When selected, it opens the Mac App Store on the "`Updates`" tab, as shown in the screenshot above. This window shows the updates that are available, and both the Apple system updates and any applications that had been downloaded and installed via the Mac App Store.



The Apple menu contains the item "`Software Update...`". When selected, it opens the Mac App Store on the "`Updates`" tab, as shown in the screenshot above. This window shows the updates that are available, and both the Apple system updates and any applications that had been downloaded and installed via the Mac App Store.

On 10.14+, the Software Update process has been integrated into System Preferences as shown in the lower right screenshot.

macOS Software Update (10.8+):
System Updates: /Library/Preferences/com.apple.SoftwareUpdate.plist
3rd Party 10.14: ~/Library/Caches/com.apple.appstoreagent/updates.plist

LastResultCode	Number	0
LastAttemptSystemVersion	String	10.14.4 (18E226)
SkipLocalCDN	Boolean	NO
LastUpdatesAvailable	Number	2
LastRecommendedUpdatesAvailable	Number	2
LastAttemptBuildVersion	String	10.14.4 (18E226)
▼ RecommendedUpdates	Array	(2 items)
▼ Item 0	Dictionary	(4 items)
Identifier	String	macOS 10.14.5 Update
Product Key	String	041-57074
Display Name	String	macOS 10.14.5 Update
Display Version	String	
▶ Item 1	Dictionary	(4 items)
LastFullSuccessfulDate	Date	Jun 23, 2019 at 8:54:28 PM
▶ PrimaryLanguages	Array	(2 items)
LastSessionSuccessful	Boolean	YES
LastBackgroundSuccessfulDate	Date	Jun 23, 2019 at 8:58:13 PM
LastSuccessfulDate	Date	Jun 23, 2019 at 8:54:28 PM

▼ availableUpdates	Array	(0 items)
▼ completedUpdates	Array	(2 items)
▶ Item 0	Dictionary	(21 items)
▼ Item 1	Dictionary	(21 items)
title	String	OneDrive
▶ store-offers	Dictionary	(1 item)
url	String	https://apps.apple.com/us/app/onedrive/id823766827?mt=12
▶ version-external-identifiers	Array	(82 items)
current-version	String	19.070.2410
bundle-id	String	com.microsoft.OneDrive-mac
release-notes	String	Thank you for using OneDrive. We are always looking to update
installDate	Date	Jun 23, 2019 at 7:43:46 PM
item-id	Number	823,766,827
version	String	19.070.2410
type	String	link
preflight	String	https://osxapps.itunes.apple.com/itunes-assets/Purple123/v4/
version-external-identifier	Number	831,647,417
updateState	Number	1
▶ artwork-urls	Array	(3 items)
url-page-type	String	software-update
link-type	String	software-update
artist-name	String	Microsoft Corporation
▶ rating	Dictionary	(5 items)
release-date	Date	Jun 13, 2019 at 5:14:01 PM
description	String	Keep your files protected and accessible on all your devices wit

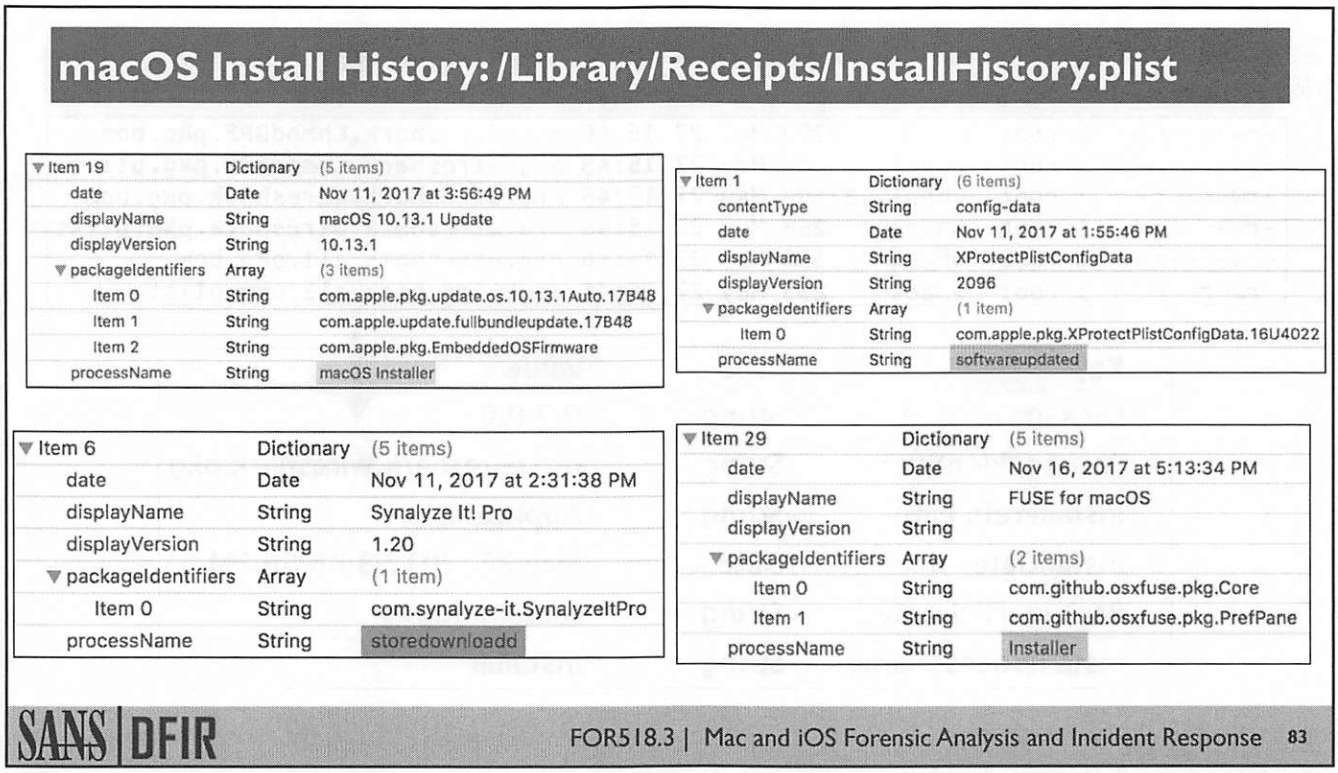
The `com.appleSoftwareUpdate.plist` property list for 10.8+ systems contains additional data. 10.14 introduced a separate plist for 3rd party updates, updates.plist in the User preferences.

This property list contains data such as when updates were last checked for (including in the background, non-user initiated), how many updates are available, and specific recommended updates.

The key `LastBackgroundSuccessfulDate` contains the timestamp of the last time updates were checked for and/or installed in the background.

LastResultCode	Number	0
LastAttemptSystemVersion	String	10.14.4 (18E226)
SkipLocalCDN	Boolean	NO
LastUpdatesAvailable	Number	2
LastRecommendedUpdatesAvailable	Number	2
LastAttemptBuildVersion	String	10.14.4 (18E226)
▼ RecommendedUpdates	Array	(2 items)
▼ Item 0	Dictionary	(4 items)
Identifier	String	macOS 10.14.5 Update
Product Key	String	041-57074
Display Name	String	macOS 10.14.5 Update
Display Version	String	
▶ Item 1	Dictionary	(4 items)
LastFullSuccessfulDate	Date	Jun 23, 2019 at 8:54:28 PM
▶ PrimaryLanguages	Array	(2 items)
LastSessionSuccessful	Boolean	YES
LastBackgroundSuccessfulDate	Date	Jun 23, 2019 at 8:58:13 PM
LastSuccessfulDate	Date	Jun 23, 2019 at 8:54:28 PM

▼ availableUpdates	Array	(0 items)
▼ completedUpdates	Array	(2 items)
▶ Item 0	Dictionary	(21 items)
▼ Item 1	Dictionary	(21 items)
title	String	OneDrive
▶ store-offers	Dictionary	(1 item)
url	String	https://apps.apple.com/us/app/onedrive/id823766827?mt=12
▶ version-external-identifiers	Array	(52 items)
current-version	String	19.070.2410
bundle-id	String	com.microsoft.OneDrive-mac
release-notes	String	Thank you for using OneDrive. We are always looking to update
installDate	Date	Jun 23, 2019 at 7:43:46 PM
item-id	Number	823,766,827
version	String	19.070.2410
type	String	link
preflight	String	https://osxapps.itunes.apple.com/itunes-assets/Purple123/v4/
version-external-identifier	Number	831,647,417
updateState	Number	1
▶ artwork-urls	Array	(3 items)
url-page-type	String	software-update
link-type	String	software-update
artist-name	String	Microsoft Corporation
▶ rating	Dictionary	(5 items)
release-date	Date	Jun 13, 2019 at 5:14:01 PM
description	String	Keep your files protected and accessible on all your devices wi



The `InstallHistory.plist` property list located in `/Library/Receipts/` contains a software install history that includes the timestamp, software package name, and what process was used to install the software.

macOS Installer = System OS Installer/Updater
softwareupdated or "Software Update" = System/Security Updates
storedownloadd = App Store Installs
Installer = External Installers

macOS Receipt Files: /var/db/receipts/

```
-rw-r--r--  1 root  wheel  35290 May 27 15:46 org.wireshark.ChmodBPF.pkg.bom
-rw-r--r--  1 root  wheel    260 May 27 15:46 org.wireshark.ChmodBPF.pkg.plist
-rw-r--r--  1 root  wheel  62594 May 27 15:46 org.wireshark.Wireshark.pkg.bom
-rw-r--r--  1 root  wheel    256 May 27 15:46 org.wireshark.Wireshark.pkg.plist
-rw-r--r--  1 root  wheel  35138 May 27 15:46 org.wireshark.cli.pkg.bom
-rw-r--r--  1 root  wheel    255 May 27 15:46 org.wireshark.cli.pkg.plist
```

Key	Type	Value
PackageVersion	String	0.0.0.0
PackageIdentifier	String	org.wireshark.Wireshark.pkg
InstallPrefixPath	String	Applications
InstallDate	Date	May 27, 2012 3:46:56 PM
PackageFileName	String	wireshark.pkg
InstallProcessName	String	Installer

The same information can be found in a slightly different format in the `/var/db/receipts` directory. Each software package install has a `.bom` and a `.plist` file. The property list file also contains the software install timestamp, package name, and the installer process.

The `.bom` file is the Mac OS X Bill of Materials (BOM) file that contains a list of files and metadata for the installed application. These can be viewed with the command `lsbom`, as shown on the next slide.

macOS Receipt Files: BOM Files (lsbom) /var/db/receipts/*.bom

```
nibble:receipts sledwards$ lsbom com.adobe.pkg.FlashPlayer.bom
.       40755   502/20
./Library       41775   0/80
./Library/Application Support   40775   0/80
./Library/Application Support/Adobe     40775   0/80
./Library/Application Support/Adobe/Flash Player Install Manager        40755   0/80
./Library/Application Support/Adobe/Flash Player Install Manager/fpsaud 100744  0/80    59248   3932184723
./Library/Internet Plug-Ins     40775   0/80
./Library/Internet Plug-Ins/Flash Player.plugin.lzma    100664  0/80    17110174        752483422
./Library/Internet Plug-Ins/flashplayer.xpt     100664  0/80    856     1969355171
./Library/LaunchDaemons 40755   0/0
./Library/LaunchDaemons/com.adobe.fpsaud.plist  100644  0/0     462     1274181950
./Library/PreferencePanes       40755   0/0
./Library/PreferencePanes/Flash Player.prefPane 40775   0/0
./Library/PreferencePanes/Flash Player.prefPane/Contents        40775   0/0
./Library/PreferencePanes/Flash Player.prefPane/Contents/Info.plist     100664  0/0     827     894812453
./Library/PreferencePanes/Flash Player.prefPane/Contents/MacOS  40775   0/0
./Library/PreferencePanes/Flash Player.prefPane/Contents/MacOS/Flash Player     100775  0/0     1212576 1112384951
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources      40775   0/0
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/FlashPlayerPreferences.nib   100664  0/0     95850   2846781901
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/FlashPlayerPreferences.png   100664  0/0     1144    1473368544
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/FlashPlayerPreferences.searchTerms   100664  0/0     1466    2994979473
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/InfoPlist.strings    100664  0/0     244     169668188
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/minusSign.png        100664  0/0     160     2182779873
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/pencilIcon.png       100664  0/0     380     2776592378
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/plusSign.png 100664  0/0     278     162556927
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/version.plist        100664  0/0     185     4240636033
```


The screenshot above shows an example of the output from the command `lsbom`. This shows the various files associated with the BOM file `com.adobe.pkg.FlashPlayer.bom`.

Default output contains:

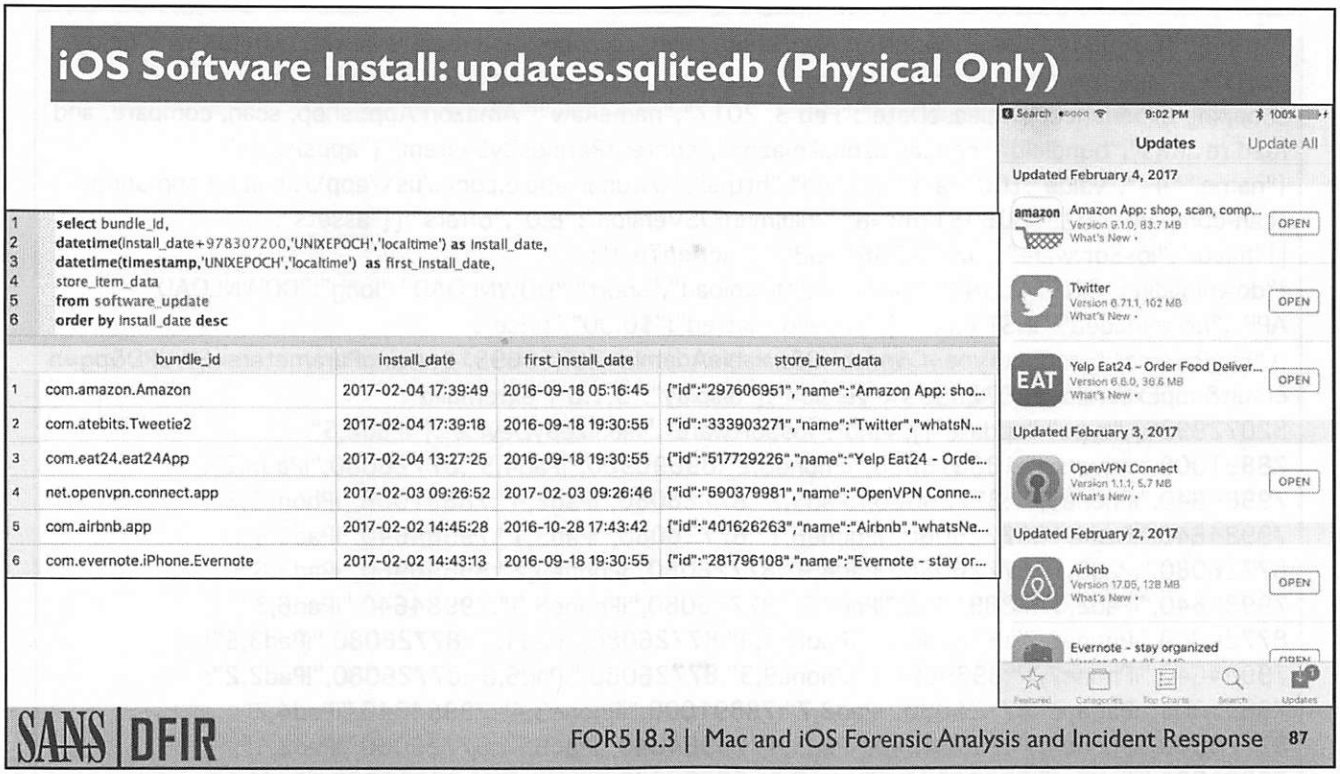
- File path
- Mode (octal)
- UID/GID

Each type of file will contain different information in the BOM file.

- If the file is a plain file, the UID/GID is followed by the file size and CRC checksum.
- If the file is a symbolic link, the UID/GID is followed by the size and CRC checksum of the link path and the link path.
- If the file is a device file, the UID/GID is followed by device number.

Reference:
Man Page for `lsbom`

```
nibble:receipts sledwards$ lsbom com.adobe.pkg.FlashPlayer.bom
.       40755   502/20
./Library       41775   0/80
./Library/Application Support   40775   0/80
./Library/Application Support/Adobe     40775   0/80
./Library/Application Support/Adobe/Flash Player Install Manager        40755   0/80
./Library/Application Support/Adobe/Flash Player Install Manager/fpsaud 100744  0/80    59248   3932184723
./Library/Internet Plug-Ins     40775   0/80
./Library/Internet Plug-Ins/Flash Player.plugin.lzma    100664  0/80    17110174        752483422
./Library/Internet Plug-Ins/flashplayer.xpt     100664  0/80    856     1969355171
./Library/LaunchDaemons 40755   0/0
./Library/LaunchDaemons/com.adobe.fpsaud.plist  100644  0/0     462     1274181950
./Library/PreferencePanes       40755   0/0
./Library/PreferencePanes/Flash Player.prefPane 40775   0/0
./Library/PreferencePanes/Flash Player.prefPane/Contents        40775   0/0
./Library/PreferencePanes/Flash Player.prefPane/Contents/Info.plist     100664  0/0     827     894812453
./Library/PreferencePanes/Flash Player.prefPane/Contents/MacOS  40775   0/0
./Library/PreferencePanes/Flash Player.prefPane/Contents/MacOS/Flash Player     100775  0/0     1212576 1112384951
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources      40775   0/0
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/FlashPlayerPreferences.nib   100664  0/0     95850   2846781901
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/FlashPlayerPreferences.png   100664  0/0     1144    1473368544
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/FlashPlayerPreferences.searchTerms   100664  0/0     1466    2994979473
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/InfoPlist.strings    100664  0/0     244     169668188
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/minusSign.png        100664  0/0     160     2182779873
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/pencilIcon.png       100664  0/0     380     2776592378
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/plusSign.png 100664  0/0     278     162556927
./Library/PreferencePanes/Flash Player.prefPane/Contents/Resources/version.plist        100664  0/0     185     4240636033
```



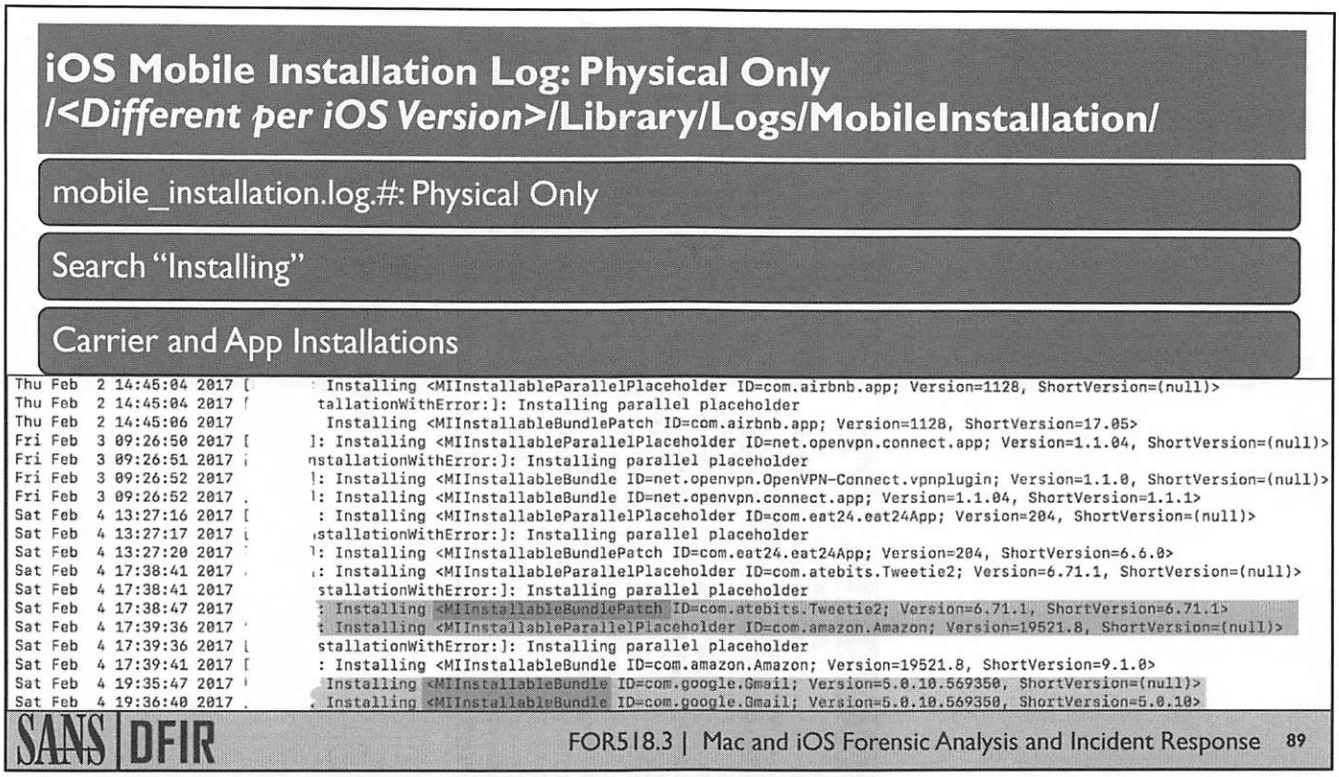
The closest database to this information is the `updates.sqlitedb` SQLite database located in `/private/var/containers/Data/System/<GUID for com.apple.appstored>/Documents/updates.sqlitedb` and is only available on physical acquisitions. This database does not appear to store all install app timestamps; however it will show apps that have updated and what appears to be their original install time. If an app has been installed but not updated, it will not show up in this database.

There is also `/private/var/mobile/Library/com.apple.itunesstored/updates.sqlitedb` that is available in physical and backups; however, I have found this database to be unreliable and misleading.

```
select
bundle_id,
datetime(install_date+978307200,'UNIXEPOCH','localtime') as install_date,
datetime(timestamp,'UNIXEPOCH','localtime') as first_install_date,
store_item_data
from software_update
order by install_date desc
```

This query extracts the bundle identifier for the app, along with the install date for the update (`install_date`) and what appears to be the original app install date (`first_install_date` as compared with `mobile_installation` logs). The `store_item_data` column contains App Store details about the application, as shown in the screenshot on the next page.

{"id":"297606951","name":"Amazon App: shop, scan, compare, and read reviews","whatsNew":"Enjoy our latest update where we have fixed some bugs and improved our app to provide you a seamless shopping experience.","releaseDate":"Feb 3, 2017","nameRaw":"Amazon App: shop, scan, compare, and read reviews","bundleId":"com.amazon.Amazon","contentRatingsBySystem":{"appsApple":{"name":"4+","value":100,"rank":1}},"url":"https:\/\/itunes.apple.com\/us\/app\/amazon-app-shop-scan-compare\/id297606951?mt=8","minimumOSVersion":"8.0","offers":[{"assets":[{"flavor":"iosSoftware","size":138618880}],"actionText":{"downloading":"INSTALLING","medium":"Download","short":"DOWNLOAD","long":"DOWNLOAD APP","downloaded":"INSTALLED"},"priceFormatted":"$0.00","price":0,"buyParams":"productType=C&price=0&salableAdamId=297606951&pricingParameters=SWUPD&pg=default&appExtVrsId=820729939","version":{"display":"9.1.0","externalId":820729939},"type":"update"}],"kind":"iosSoftware","fileSizeByDevice":{"iPad2,5":78891008,"universal":138618880,"iPhone8,2":89896960,"iPad4,5":87726080,"iPad3,2":79984640,"iPhone9,1":87726080,"iPad5,2":87726080,"iPad2,4":78891008,"iPhone5,2":79984640,"iPad4,4":87726080,"iPhone6,1":87726080,"iPad3,1":79984640,"iPad6,4":87726080,"iPad5,1":87726080,"iPad4,9":87726080,"iPhone9,2":89896960,"iPad3,6":79984640,"iPad2,3":78891008,"iPad4,3":87726080,"iPhone5,3":79984640,"iPad6,3":87726080,"iPhone6,2":87726080,"iPhone8,4":87726080,"iPad4,8":87726080,"iPad3,5":79984640,"iPhone7,1":89896960,"iPhone9,3":87726080,"iPad6,8":87726080,"iPad2,2":78891008,"iPad4,2":87726080,"iPad2,7":78891008,"iPhone5,4":79984640,"iPad4,7":87726080,"iPhone4,1":79984640,"iPad3,4":79984640,"iPad6,7":87726080,"iPad2,1":78891008,"iPad5,4":87726080,"iPhone7,2":87726080,"iPhone9,4":89896960,"iPad4,1":87726080,"iPhone8,1":87726080,"iPad2,6":78891008,"iPad4,6":87726080,"iPad3,3":79984640,"iPod5,1":79984640,"iPad5,3":87726080,"iPod7,1":87726080,"iPhone5,1":79984640},"artwork":{"height":1024,"textColor3":"333333","textColor4":"3a5d74","textColor2":"093551","supportsLayeredImage":false,"width":1024,"textColor1":"000000","bgColor":"ffffff","url":"http:\/\/is1.mzstatic.com\/image\/thumb\/Purple122\/v4\/38\/e4\/2e\/38e42e78-c917-3653-2a22-33aa462d8f1d\/source\/{w}x{h}bb.{f}"},"artistName":"AMZN Mobile LLC"}



Directory Paths (Physical Device):
iOS 6: `/mobile/Library/Logs/MobileInstallation/`
iOS 7–9: `/private/var/mobile/Library/Logs/MobileInstallation/`
iOS 10: `/private/var/installd/Library/Logs/MobileInstallation/`

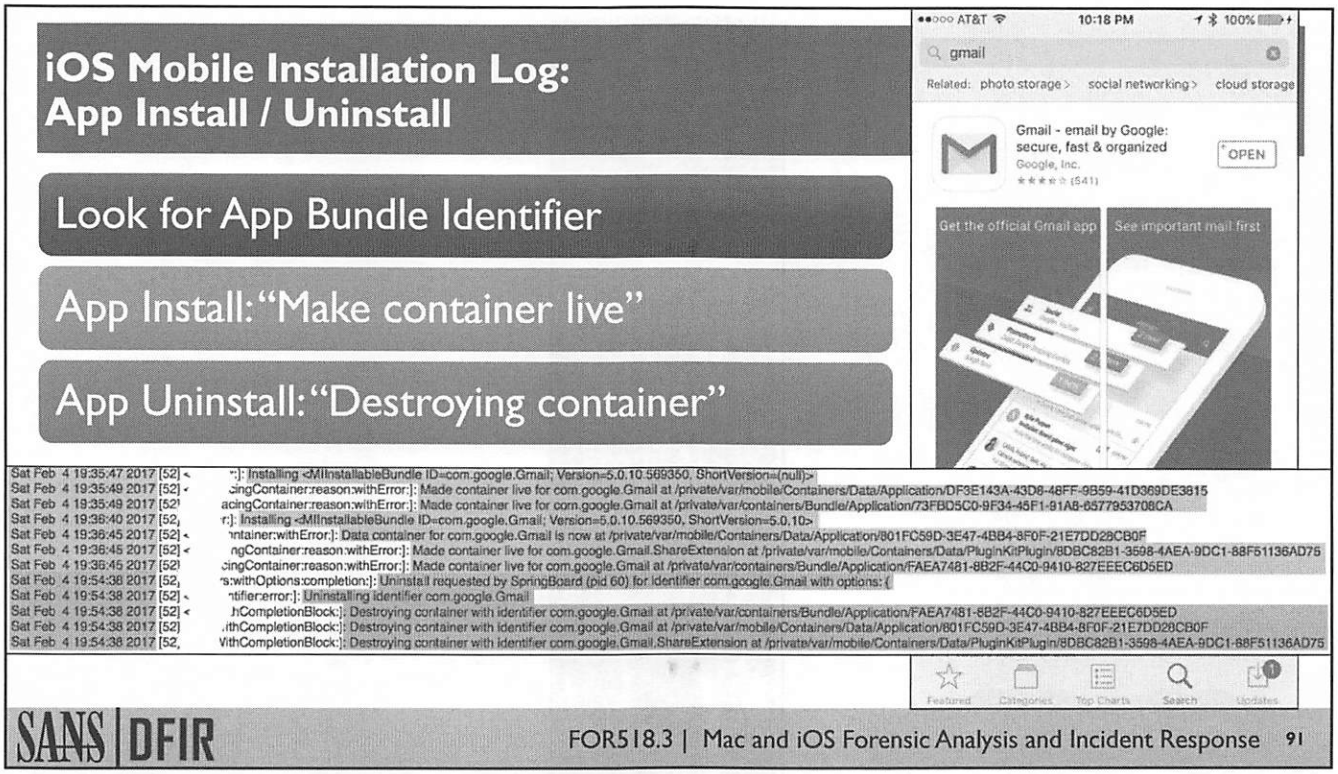
The `mobile_installation.log.#` log files located in the `/MobileInstallation/` directory contains a historical view of when apps were installed and carrier bundles were installed. How long these entries are kept depends on the version of iOS being used. The older the iOS version, the longer these entries are kept. On iOS 10, it appears to be kept for around one month.

When an app is installed, it is recorded in this log using the application bundle identifier (reverse DNS format name). For example, Google Gmail (`com.google.Gmail`) was installed on February 4, 2017, at 19:36:40 (local device time). It will also show the application version information in the message.

From the entries highlighted in the screenshot above, an analyst can tell if it is an app update or a new app by looking at the message after the "`Installing`" keyword.

`MIInstallableBundlePatch`: App Update
`MIInstallerBundle`: New App Install

```
Thu Feb  2 14:45:04 2017 [        : Installing <MIInstallableParallelPlaceholder ID=com.airbnb.app; Version=1128, ShortVersion=(null)>
Thu Feb  2 14:45:04 2017 [        tallationWithError:]: Installing parallel placeholder
Thu Feb  2 14:45:06 2017          : Installing <MIInstallableBundlePatch ID=com.airbnb.app; Version=1128, ShortVersion=17.05>
Fri Feb  3 09:26:50 2017 [        ]: Installing <MIInstallableParallelPlaceholder ID=net.openvpn.connect.app; Version=1.1.04, ShortVersion=(null)>
Fri Feb  3 09:26:51 2017 [        nstallationWithError:]: Installing parallel placeholder
Fri Feb  3 09:26:52 2017          ]: Installing <MIInstallableBundle ID=net.openvpn.OpenVPN-Connect.vpnplugin; Version=1.1.0, ShortVersion=(null)>
Fri Feb  3 09:26:52 2017 .        ]: Installing <MIInstallableBundle ID=net.openvpn.connect.app; Version=1.1.04, ShortVersion=1.1.1>
Sat Feb  4 13:27:16 2017 [        : Installing <MIInstallableParallelPlaceholder ID=com.eat24.eat24App; Version=204, ShortVersion=(null)>
Sat Feb  4 13:27:17 2017 [        nstallationWithError:]: Installing parallel placeholder
Sat Feb  4 13:27:20 2017 [        ]: Installing <MIInstallableBundlePatch ID=com.eat24.eat24App; Version=204, ShortVersion=6.6.0>
Sat Feb  4 17:38:41 2017 .        ]: Installing <MIInstallableParallelPlaceholder ID=com.atebits.Tweetie2; Version=6.71.1, ShortVersion=(null)>
Sat Feb  4 17:38:41 2017          stallationWithError:]: Installing parallel placeholder
Sat Feb  4 17:38:47 2017          : Installing <MIInstallableBundlePatch ID=com.atebits.Tweetie2; Version=6.71.1, ShortVersion=6.71.1>
Sat Feb  4 17:39:36 2017 [        : Installing <MIInstallableParallelPlaceholder ID=com.amazon.Amazon; Version=19521.8, ShortVersion=(null)>
Sat Feb  4 17:39:36 2017 [        stallationWithError:]: Installing parallel placeholder
Sat Feb  4 17:39:41 2017 [        : Installing <MIInstallableBundle ID=com.amazon.Amazon; Version=19521.8, ShortVersion=9.1.0>
Sat Feb  4 19:35:47 2017 [         Installing <MIInstallableBundle ID=com.google.Gmail; Version=5.0.10.569350, ShortVersion=(null)>
Sat Feb  4 19:36:40 2017 .        . Installing <MIInstallableBundle ID=com.google.Gmail; Version=5.0.10.569350, ShortVersion=5.0.10>
```



While application installs can be observed in the Mobile Installation logs, uninstalls can also be seen. The example screenshot above is the same Google Gmail application being installed, then later uninstalled.

In the entries highlighted in green (at the top), the Gmail app is being installed on February 4 at 19:36. The various entries show the different containers and directory paths being created to store the various pieces of the app data.

Upon uninstall, the entries highlighted in red show the same containers being "destroyed". This particular app uninstall was performed by selecting the Gmail app via SpringBoard until it "wiggles" and selecting the "X" to remove the app.

```
Sat Feb  4 19:35:47 2017 [52] <          r:]: Installing <MIInstallableBundle ID=com.google.Gmail; Version=5.0.10.569350, ShortVersion=(null)>
Sat Feb  4 19:35:49 2017 [52] <          cingContainer:reason:withError:]: Made container live for com.google.Gmail at /private/var/mobile/Containers/Data/Application/DF3E143A-43D8-48FF-9B59-41D369DE3815
Sat Feb  4 19:35:49 2017 [52]          acingContainer:reason:withError:]: Made container live for com.google.Gmail at /private/var/containers/Bundle/Application/73FBD5C0-9F34-45F1-91A8-6577953708CA
Sat Feb  4 19:36:40 2017 [52,          r:]: Installing <MIInstallableBundle ID=com.google.Gmail; Version=5.0.10.569350, ShortVersion=5.0.10>
Sat Feb  4 19:36:45 2017 [52] <          ntainer:withError:]: Data container for com.google.Gmail is now at /private/var/mobile/Containers/Data/Application/801FC59D-3E47-4BB4-8F0F-21E7DD28CB0F
Sat Feb  4 19:36:45 2017 [52] <          ngContainer:reason:withError:]: Made container live for com.google.Gmail.ShareExtension at /private/var/mobile/Containers/Data/PluginKitPlugin/8DBC82B1-3598-4AEA-9DC1-88F51136AD75
Sat Feb  4 19:36:45 2017 [52]          cingContainer:reason:withError:]: Made container live for com.google.Gmail at /private/var/containers/Bundle/Application/FAEA7481-8B2F-44C0-9410-827EEEC6D5ED
Sat Feb  4 19:54:38 2017 [52,          s:withOptions:completion:]: Uninstall requested by SpringBoard (pid 60) for identifier com.google.Gmail with options: {
Sat Feb  4 19:54:38 2017 [52] <          ntifier:error:]: Uninstalling identifier com.google.Gmail
Sat Feb  4 19:54:38 2017 [52] <          hCompletionBlock:]: Destroying container with identifier com.google.Gmail at /private/var/containers/Bundle/Application/FAEA7481-8B2F-44C0-9410-827EEEC6D5ED
Sat Feb  4 19:54:38 2017 [52]          ithCompletionBlock:]: Destroying container with identifier com.google.Gmail at /private/var/mobile/Containers/Data/Application/801FC59D-3E47-4BB4-8F0F-21E7DD28CB0F
Sat Feb  4 19:54:38 2017 [52,          VithCompletionBlock:]: Destroying container with identifier com.google.Gmail.ShareExtension at /private/var/mobile/Containers/Data/PluginKitPlugin/8DBC82B1-3598-4AEA-9DC1-88F51136AD75
```

iOS App Uninstall (Physical Only)
/private/var/installd/Library/MobileInstallation/UninstalledApplications.plist

RunKeeperPro	Date	Sep 18, 2016, 10:06:10 PM
com.H443NM7F8H.CBSNews	Date	Sep 17, 2016, 12:43:19 AM
com.HBO.HBO	Date	Sep 23, 2016, 10:26:52 PM
com.aetn.aetv.ios.watch	Date	Sep 17, 2016, 12:44:33 AM
com.aol.aim	Date	Sep 18, 2016, 10:03:17 PM
com.bakodo.BakodoScanner	Date	Sep 18, 2016, 10:02:23 PM
com.brickoapps.CALC-	Date	Sep 18, 2016, 10:03:31 PM
com.cbsvideo.app	Date	Sep 17, 2016, 12:43:15 AM
com.chrisballinger.ChatSecure	Date	Sep 18, 2016, 10:02:21 PM
com.clickgamer.AngryBirdsLite	Date	Sep 23, 2016, 10:29:24 PM
com.fox.now	Date	Sep 17, 2016, 12:43:13 AM
com.fxnetworks.ios.FXNOW	Date	Sep 17, 2016, 12:43:28 AM
com.google.Gmail	Date	Feb 4, 2017, 7:54:38 PM
com.ihandysoft.barcode.qr.free	Date	Sep 18, 2016, 10:02:25 PM
com.natgeo.tv	Date	Sep 17, 2016, 12:43:09 AM
com.nbcuni.nbc.portal	Date	Sep 17, 2016, 12:43:54 AM

SANS | DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 93

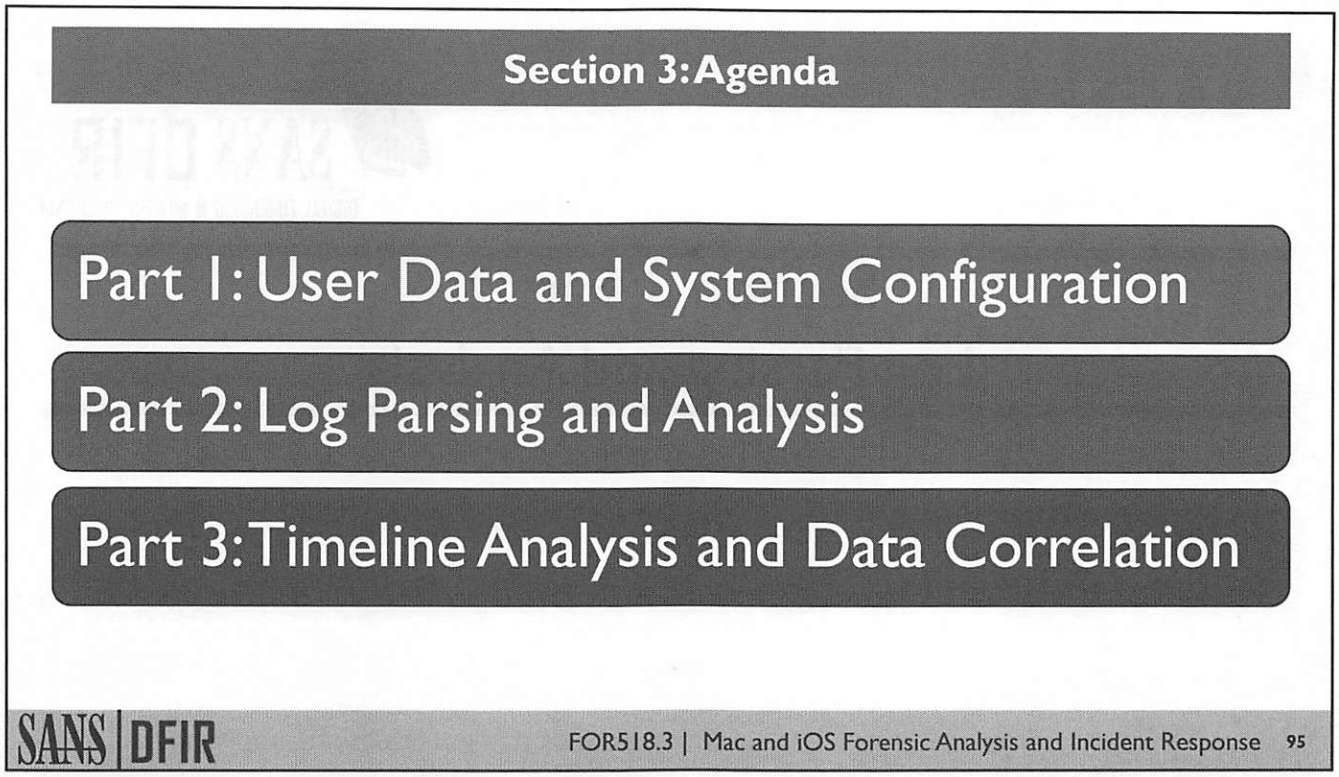
File Paths (Physical Device):
iOS 7–9:
`/private/var/mobile/Library/MobileInstallation/UninstalledApplications.plist`
iOS 10:
`/private/var/installd/Library/MobileInstallation/UninstalledApplications.plist`

Another file that can be viewed to get a quick overview of uninstalled applications is the `UninstalledApplications.plist` file. This file keeps track of uninstalled applications by bundle identifier and the date of uninstall.

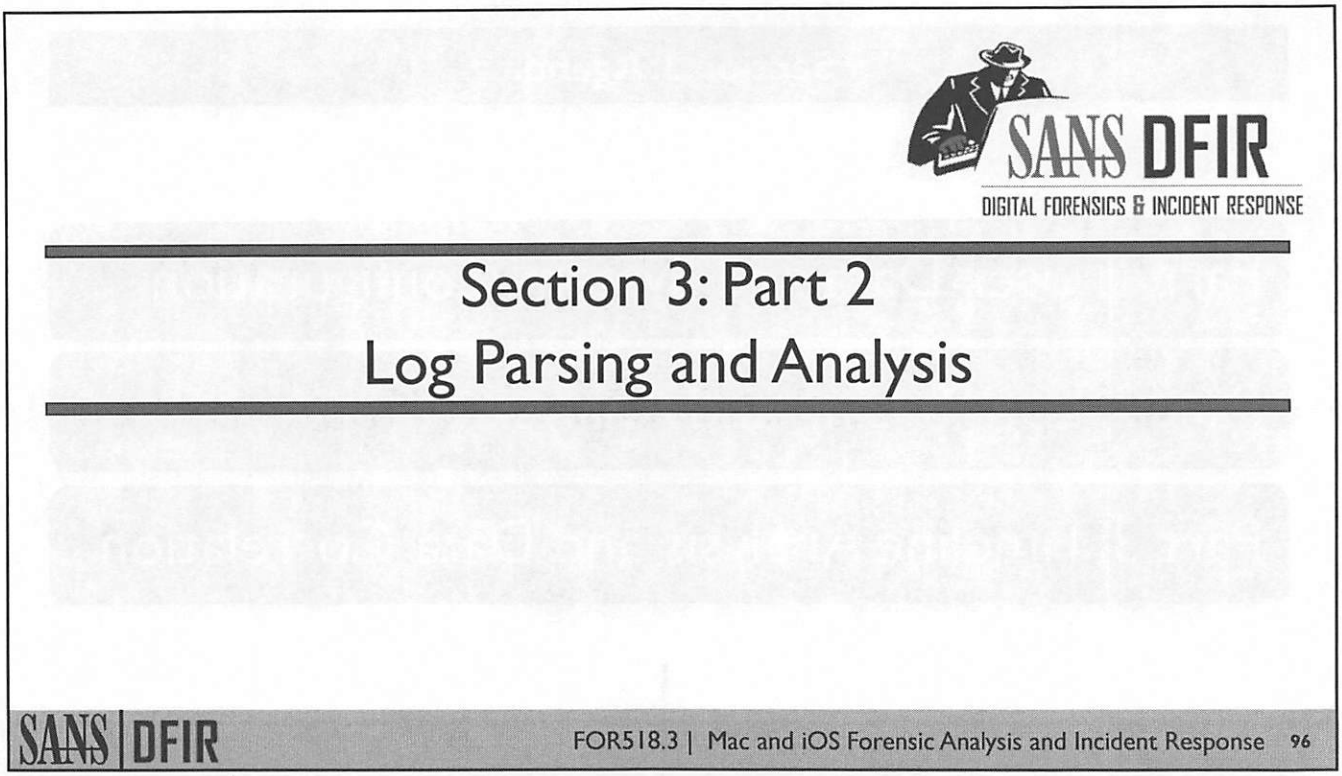
Once an application has been deleted, the application's data is no longer available for forensic analysis.

Lab 3.2
User Data and System Configuration – Part II

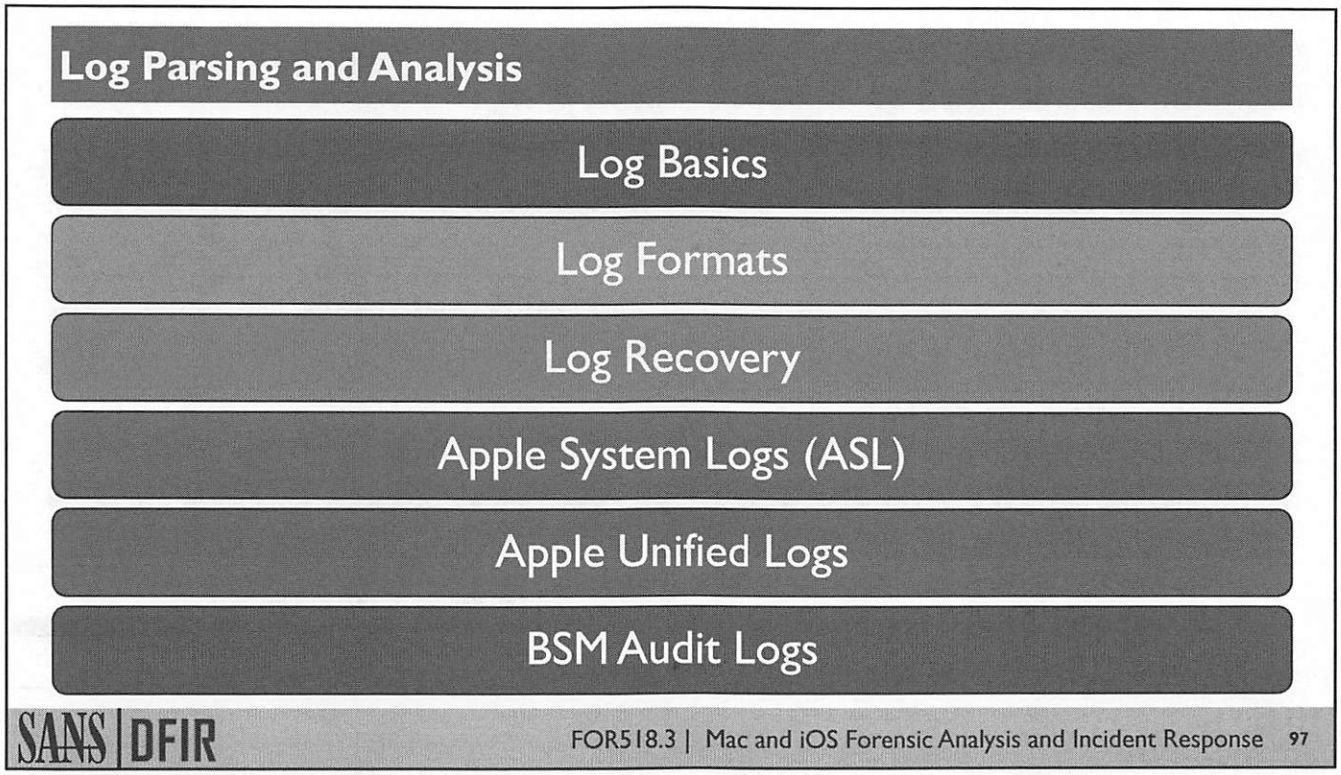
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.

Log Normalization

Correlate data in a single system or across multiple systems

Must know "originating" time zone for system

Timestamp Storage

- Apple System Log = UTC
- Most other logs (/var/log, ~/Library/Logs/) = local system time

Timestamp Output

- ASL Logs: praudit may output to local system time
- Use export TZ=UTC command
- Temporarily change time zone of terminal window

```
[Sarahs-MacBook-Pro-6:~ oompa$ date
 Fri May  4 15:07:23 EDT 2018
[Sarahs-MacBook-Pro-6:~ oompa$ export TZ="UTC"
[Sarahs-MacBook-Pro-6:~ oompa$ date
 Fri May  4 19:07:35 UTC 2018
```

SANS | DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 98

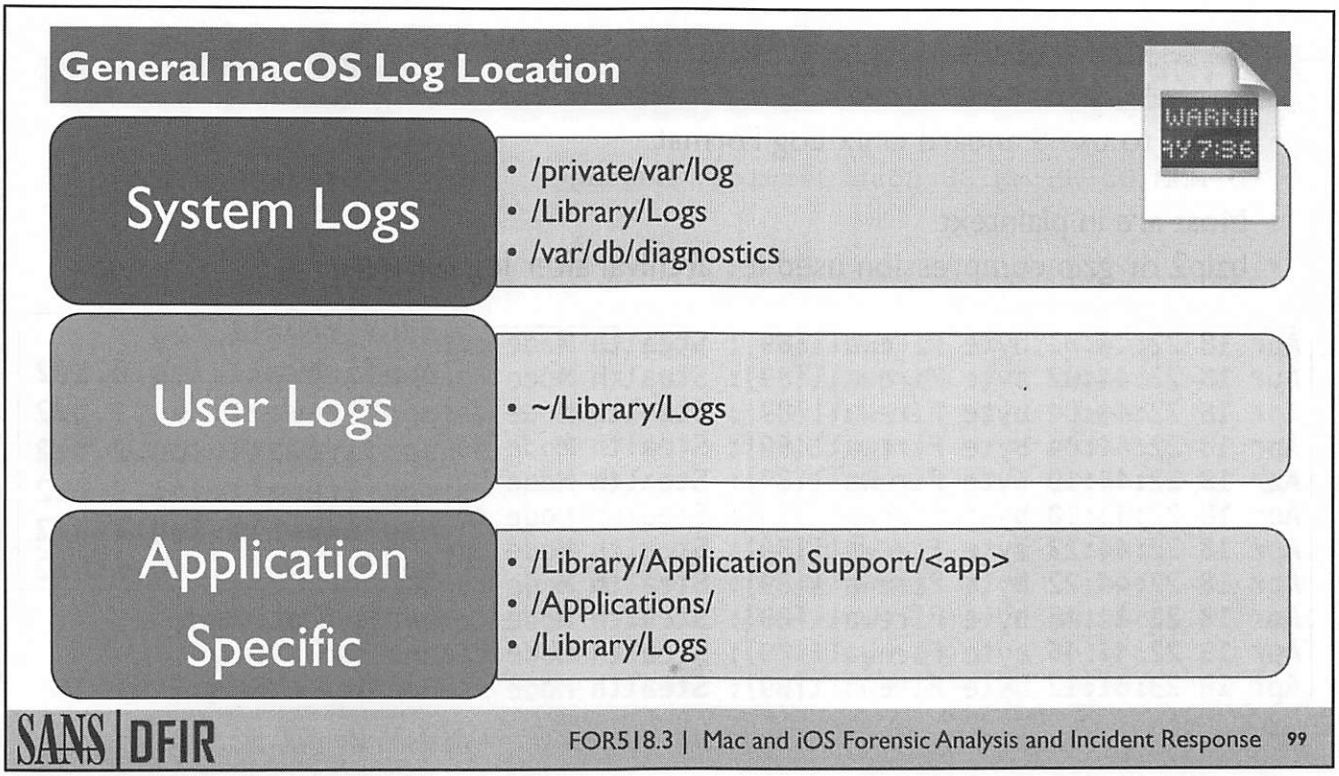
Analysts often have to correlate information across multiple systems across different time zones. To combine and understand what each system was doing at the same time, the analyst will need to normalize the log.

Each log type may store its timestamp in various formats, while some output tools will export this timestamp in a different time zone or using local system time.

One example of this is the `praudit` tool to read BSM audit logs. This tool outputs XML output with a timestamp in local system time. The analyst must know the original time zone of the system to correctly normalize the `praudit` output.

An investigator can temporarily change the time zone of the Terminal window by using the `export TZ="UTC"` command with the correct time zone (found in `/usr/share/zoneinfo/`). This time zone change will be removed once you exit out of that Terminal (login) session.

```
[Sarahs-MacBook-Pro-6:~ oompa$ date
 Fri May  4 15:07:23 EDT 2018
[Sarahs-MacBook-Pro-6:~ oompa$ export TZ="UTC"
[Sarahs-MacBook-Pro-6:~ oompa$ date
 Fri May  4 19:07:35 UTC 2018
```



There are three primary locations in OS X where logs are found.

System logs – those that have to do with the operating system – can be found in `/private/var/log`, `/Library/Logs`, and `/var/db/diagnostics`.

User-specific logs are found in each user account in their Library directory, `~/Library/Logs`.

Application logs may be found in `/Library/Application Support/` or `/Applications/` directory under the particular application.

Unix Log Basics

- Tends to use Standard Unix Log Format
  - `MMM DD HH:MM:SS Host Service: Message`
- Most are in plaintext
- bzip2 or gzip compression used for archival after log turnover

```
Apr 18 22:44:02 byte Firewall[89]: Stealth Mode co
Apr 18 22:44:02 byte Firewall[89]: Stealth Mode co
Apr 18 22:44:04 byte Firewall[89]: Stealth Mode co
Apr 18 22:44:04 byte Firewall[89]: Stealth Mode co
Apr 18 22:44:10 byte Firewall[89]: Stealth Mode co
Apr 18 22:44:10 byte Firewall[89]: Stealth Mode co
Apr 18 22:44:22 byte Firewall[89]: Stealth Mode co
Apr 18 22:44:22 byte Firewall[89]: Stealth Mode co
Apr 18 22:44:46 byte Firewall[89]: Stealth Mode connection attempt
Apr 18 22:44:46 byte Firewall[89]: Stealth Mode connection attempt
Apr 18 23:01:12 byte Firewall[89]: Stealth Mode connection attempt
```

```
appfirewall.log
appfirewall.log.0.bz2
appfirewall.log.1.bz2
appfirewall.log.2.bz2
appfirewall.log.3.bz2
appfirewall.log.4.bz2
appfirewall.log.5.bz2
```


Most of the logs found in OS X tend to use the standard Unix Log Format. This format uses a date format that does not include a year or a time zone to put context to the log data.

Most of the logs are available in plaintext format, meaning they can be read without further processing or parsing. Normal Unix operating systems will archive (rotate) log files after they grow to a certain size or are old enough (look in their associated `.conf` files in `/etc`).

Generally speaking, on 10.8 and prior systems, bzip2 compression is used to archive its log data, while on 10.9 systems, gzip is used.

Bzip2 or Gzip Decompression

- Use `bzcat` or `gzcat` on macOS
  - (oldest -> newest)
  - **Bzip2**: system.log.7.bz2 -> system.log.0.bz2
  - **Gzip**: system.log.7.gz -> system.log.0.gz

```
1.bzcat system.log.{7..0}.bz2 >
  system_all.log
2.cat system.log >> system_all.log
```


If required, the archived logs can be decompressed using the command `bzcat`. To create a comprehensive log file with the entries in the correct temporal context (oldest to newest), the `bzcat` command can be used to "concatenate" the contents of the archive files. The larger the number (i.e., `system.log.7.bz2`), the older the log archive.

- 0 = newest
- 7 = oldest

The first command concatenates the contents of the archive files from oldest to newest into a file named `system_all.log`. The second command concatenates the current log file `system.log` to the `system_all.log` file to create a complete `system.log` file for the system.

Apple System Log

- Location: /private/var/log/asl/ (>10.5.6)
- syslog "replacement" (still uses syslog backend)
- View using Console.app or `syslog` command
- Binary format: "ASL DB" signature
- Log turnover: Seven days, ~one year (utmp)

```
4153 4c20 4442 0000 0000 0000 0000 0002  ASL DB........
0000 0000 0000 00f6 0000 0000 51a2 054b  ............Q..K
0000 0100 0000 0000 0003 6c3a 0000 0000  ..........l:....
0000 0000 0000 0000 0000 0000 0000 0000  ................
0000 0000 0000 0000 0000 0000 0000 0000  ................
0001 0000 007b 6861 6e64 6c65 5f77 696c  .....{handle_wil
6c5f 736c 6565 705f 6175 7468 5f61 6e64  l_sleep_auth_and
5f73 6869 656c 645f 7769 6e64 6f77 733a  _shield_windows:
2072 656c 6561 7369 6e67 2061 7574 6877   releasing authw
2030 7837 6662 3562 6663 3034 3932 3028   0x7fb5bfc04920(
3230 3030 292c 2073 6869 656c 6420 3078  2000), shield 0x
3766 6235 6262 6365 6362 3130 2832 3030  7fb5bbcecb10(200
3129 2c20 6c6f 636b 2073 7461 7465 2033  1), lock state 3
```


The Apple System Log is Apple's version of the Unix SYSLOG. After 10.5.6, the ASL data is located in the `/var/log/asl` directory in a proprietary binary format (rather than plaintext). These messages can be viewed using the `Console.app` or the syslog command-line utility.

The screenshot shows the signature for an ASL log, "`ASL DB`", which will be found in the first six bytes of each log file.

The default time-to-live (TTL) for SYSLOG messages is seven days, while the default TTL for utmp, wtmp, and lastlog messages (i.e., logon/logoff/boot/shutdown/restart) is one year (366 days or 31622400 seconds as per man `asl.conf`).

Reference:
`asl.conf` Man Page

Apple System Log: Filenames

- Filename Format: YYYY.MM.DD.[UID].[GID].asl
- BB: Best Before
- AUX: Auxiliary

```
nibble:AUX.2013.05.28 sledwards$ pwd
/var/log/asl/AUX.2013.05.28
nibble:AUX.2013.05.28 sledwards$ ls
281501  281597  281692  281790  281884
281503  281599  281698  281792  281886
281505  281604  281700  281794  281892
281511  281606  281702  281801  281894
281513  281608  281708  281803  281896
281515  281614  281710  281805  281902
281521  281616  281712  281810  281904
281523  281618  281718  281812  281906
281525  281624  281721  281814  281912
281531  281626  281723  281820  281914
```

```
May 28 23:57 2013.05.28.G80.asl
May 28 23:59 2013.05.28.U0.G80.asl
May 28 23:49 2013.05.28.U0.asl
May 28 22:15 2013.05.28.U501.asl
May 29 23:58 2013.05.29.G80.asl
May 29 23:58 2013.05.29.U0.G80.asl
May 29 22:45 2013.05.29.U0.asl
May 29 23:21 2013.05.29.U501.asl
May 30 23:57 2013.05.30.G80.asl
May 30 23:57 2013.05.30.U0.G80.asl
May 30 23:49 2013.05.30.U0.asl
May 30 22:41 2013.05.30.U501.asl
May 31 23:59 2013.05.31.G80.asl
May 31 23:59 2013.05.31.U0.G80.asl
May 31 22:52 2013.05.31.U0.asl
May 31 23:08 2013.05.31.U501.asl
Jun  1 23:59 2013.06.01.G80.asl
Jun  1 23:59 2013.06.01.U0.G80.asl
Jun  1 23:17 2013.06.01.U0.asl
Jun  1 21:45 2013.06.01.U501.asl
Jun  2 23:58 2013.06.02.G80.asl
Jun  2 23:58 2013.06.02.U0.G80.asl
Jun  2 23:06 2013.06.02.U0.asl
Jun  2 21:22 2013.06.02.U501.asl
Jun  3 20:08 2013.06.03.G80.asl
Jun  3 20:08 2013.06.03.U0.G80.asl
Jun  3 19:21 2013.06.03.U0.asl
Jun  3 10:57 2013.06.03.U200.asl
Jun  3 19:55 2013.06.03.U501.asl
May 28 23:56 AUX.2013.05.28
May 29 23:57 AUX.2013.05.29
May 30 23:56 AUX.2013.05.30
May 31 23:59 AUX.2013.05.31
Jun  1 23:58 AUX.2013.06.01
Jun  2 23:58 AUX.2013.06.02
Jun  3 20:08 AUX.2013.06.03
Mar 30 09:59 BB.2014.03.31.G80.asl
Apr 25 17:35 BB.2014.04.30.G80.asl
May 29 20:52 BB.2014.05.31.G80.asl
```

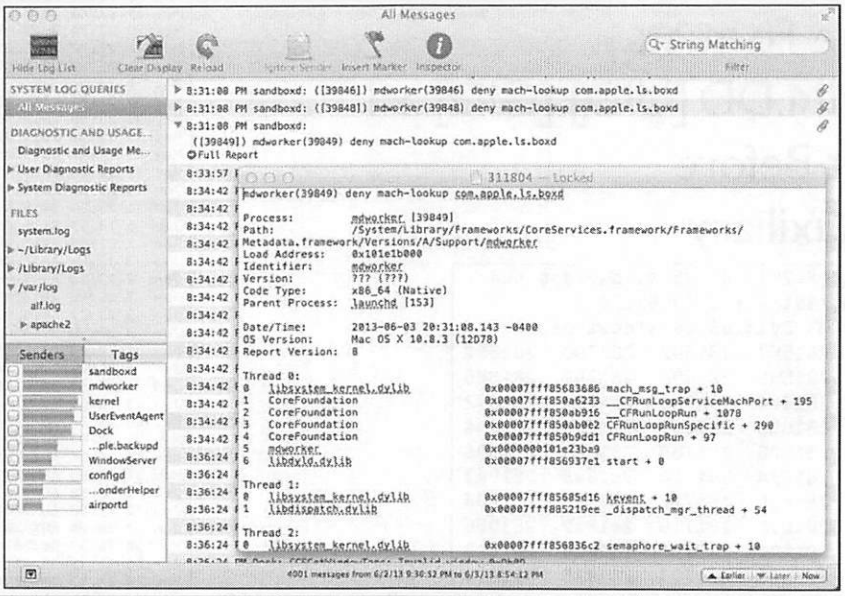
SANS | DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 103

The ASL filename format is standardized to match the month, day, and year the data was recorded. The files are also separated by user and group IDs. These logs will usually be seen for only the past seven days, after the `utmp`, `wtmp`, and `lastlog` messages get rolled into the ASL log files beginning with "`BB`", or "Best Before".

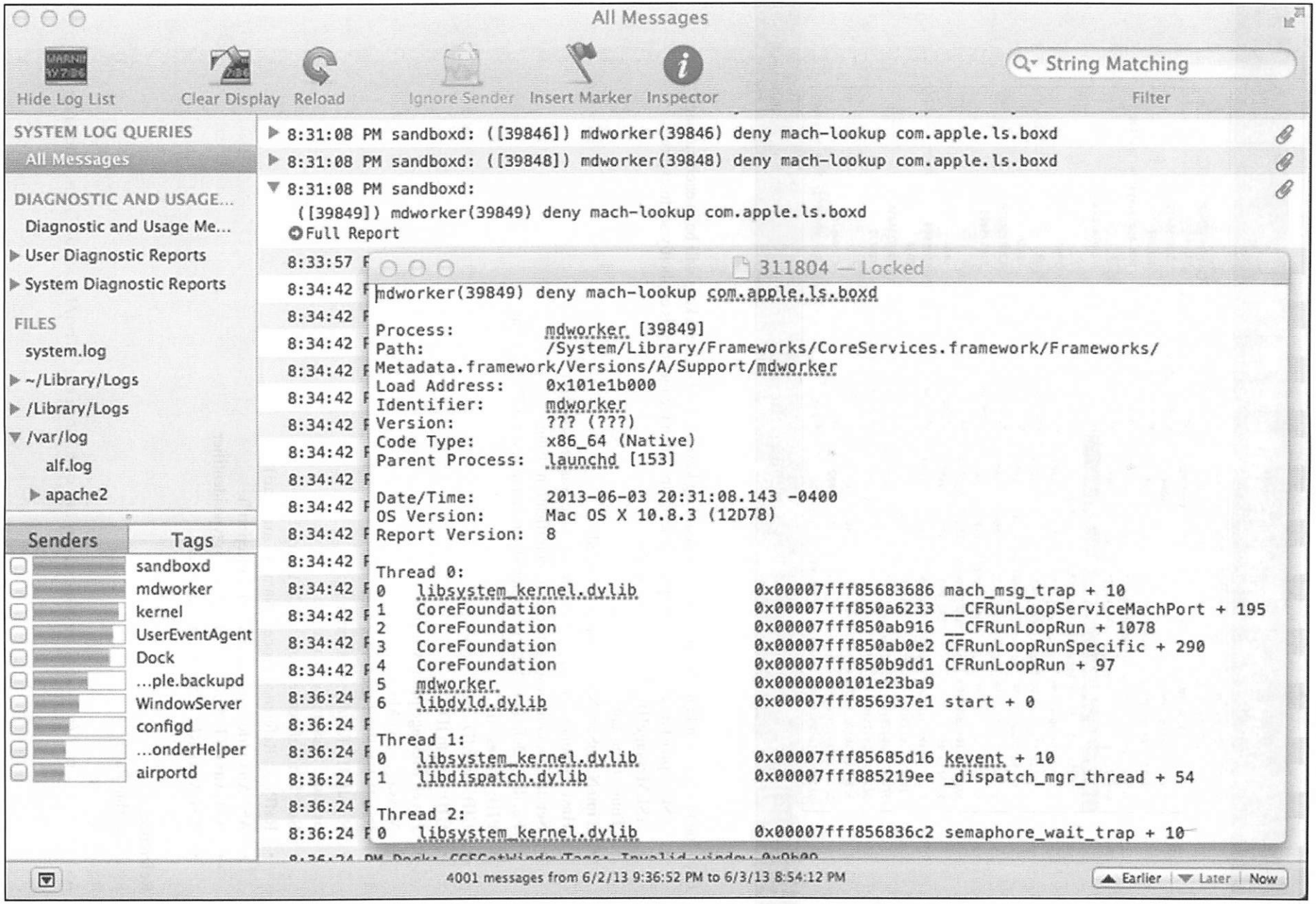
The "Best Before" log files contain the login records for the month listed in the filename. These records will be kept around for one year.

The directories starting with `AUX` or "Auxiliary" are used on 10.8+ systems. These directories contain text files comprised of additional data that is referenced by the `syslog` files for the past week. In my experience, these files show information pertaining to the `sandboxd` process or Apple Sandboxing.

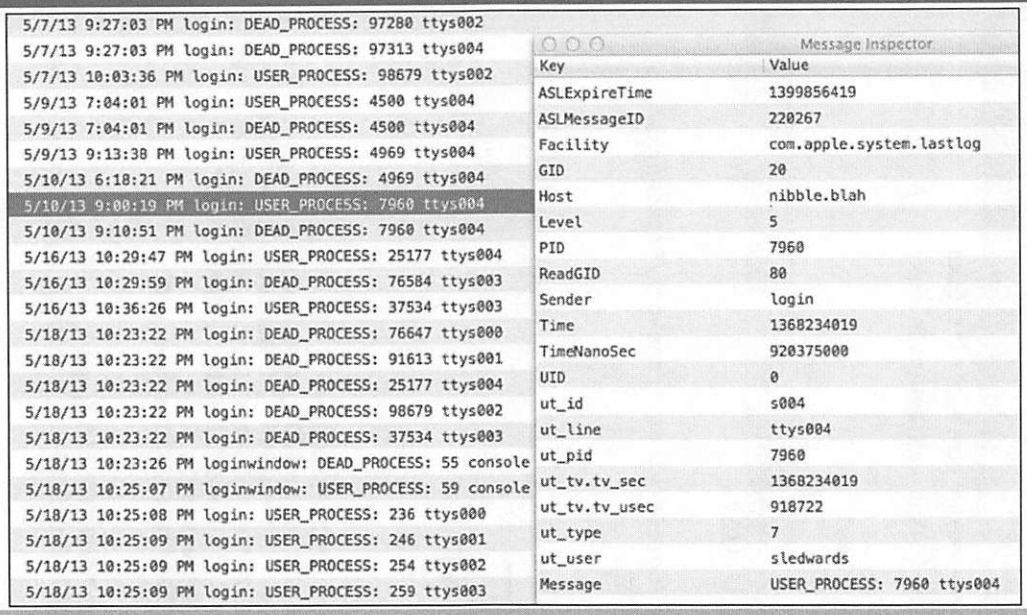
Apple System Logs: Auxiliary Files



The screenshot above shows how the auxiliary files interface with the `syslog` files. In the Console application, the messages with the paper-clip "attachment" icon contain additional information. The "Full Report" links to the additional data in the auxiliary directory.



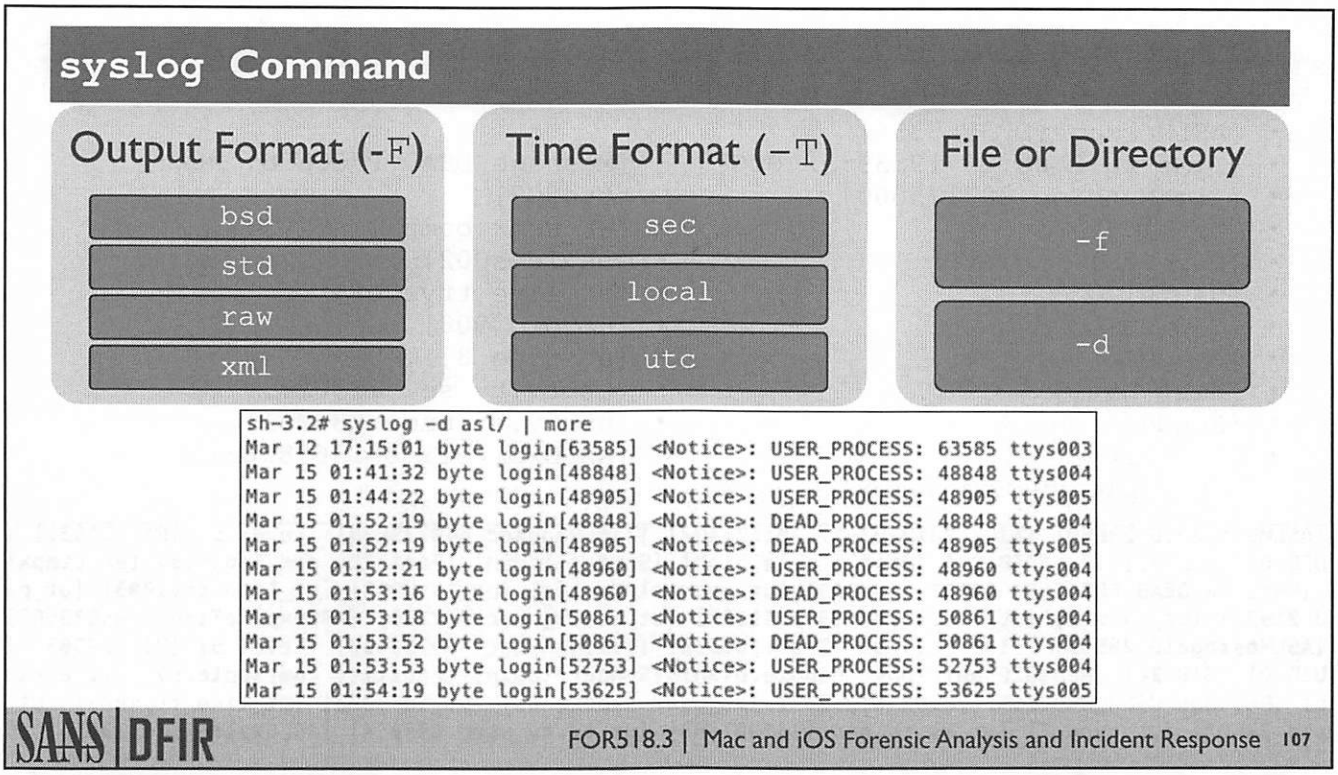
Apple System Log: Record Format



Each log message contains certain Apple System Log keys. Not all of these keys will be in each message.

- **ASLExpireTime**: Message Expire Timestamp, when the message is explicitly set to expire
- **ASLMessageID:** Message ID Number
- **Time**: Timestamp of the record
- **TimeNanoSec**: Nanoseconds recorded
- **Host**: Hostname of the system the message was recorded on
- **Sender**: Default process name or identification string
- **Facility**: Default is "`user`", otherwise noted in reverse DNS format
- **PID**: Process ID
- **UID**: User ID
- **GID**: Group ID
- **Level**: Message priority level
- **Message**: Log Message
- **ReadUID**: Read access for user
- **ReadGID**: Read access for group
- **Session**: Session by `launchd`
- **RefPID:** Reference Process ID (`launchd`)
- **RefProc**: Reference Process Name (`launchd`)
- **ASLAuxTitle**: Title (usually "Full Report")
- **ASLAuxUTI**: ASL auxiliary uniform type identifier
- **ASLAuxURL**: URL to auxiliary file

References:
`syslog` Man Page
`ASL` Man Page
`Asl.h – http://www.opensource.apple.com/source/Libc/Libc-583/include/asl.h`



A single ASL file or a directory containing multiple ASL files can be used as input into the `syslog` command.

The syslog command can be used to output the data in a variety of formats:

- `bsd`: Similar to other system logs such as `system.log`
- `std`: Same as `bsd`, and includes message priority level (default)
- `raw`: Message fields are in square brackets, in key/value format
- `xml`: XML property list

The time output can also be formatted:

- `sec`: Number of seconds since epoch
- `local`: Local time zone (default)
- `utc`: UTC format

In the example shown above, the `syslog` command is taking in the default directory of ASL files and outputting the data in the default standard format.

```
syslog -T utc -F raw -d /asl
```

- [ASLMessageID 3555356]
- [Time 2012.05.28 19:39:32 UTC]
- [TimeNanoSec 887175000]
- [Level 5]
- [PID 908]
- [UID 0]
- [GID 20]
- [ReadGID 80]
- [Host byte]
- [Sender login]
- [Facility com.apple.system.utmpx]
- [Message DEAD_PROCESS: 908 ttys002]
- [ut_user oompa]
- [ut_id s002]
- [ut_line ttys002]
- [ut_pid 908]
- [ut_type 8]
- [ut_tv.tv_sec 1338233972]
- [ut_tv.tv_usec 886961]
- [ASLExpireTime 1369856372]

```
[ASLMessageID 23869] [Time 2013-03-17 20:12:49Z] [TimeNanoSec 649773000] [Level 5] [PID 21931] [
UID 0] [GID 20] [ReadGID 80] [Host nibble.blah] [Sender login] [Facility com.apple.system.utmpx]
 [Message DEAD_PROCESS: 21931 ttys003] [ut_user sledwards] [ut_id s003] [ut_line ttys003] [ut_pi
d 21931] [ut_type 8] [ut_tv.tv_sec 1363551169] [ut_tv.tv_usec 647288] [ASLExpireTime 1395173569]
[ASLMessageID 28599] [Time 2013-03-23 00:10:53Z] [TimeNanoSec 859756000] [Level 5] [PID 28503] [
UID 0] [GID 20] [ReadGID 80] [Host nibble.blah] [Sender login] [Facility com.apple.system.lastlo
g] [Message USER_PROCESS: 28503 ttys003] [ut_user sledwards] [ut_id s003] [ut_line ttys003] [ut_
pid 28503] [ut_type 7] [ut_tv.tv_sec 1363997453] [ut_tv.tv_usec 859054] [ASLExpireTime 139561985
3]
```

The raw output for `syslog` labels each key/value pair in square brackets, as shown above. All dates are stored in Unix epoch time. The fields starting with "`ut_`" are a throwback to the `utmp` login data that has been deprecated in OS X since 10.5.

```
[ASLMessageID 23869] [Time 2013-03-17 20:12:49Z] [TimeNanoSec 649773000] [Level 5] [PID 21931] [
UID 0] [GID 20] [ReadGID 80] [Host nibble.blah] [Sender login] [Facility com.apple.system.utmpx]
 [Message DEAD_PROCESS: 21931 ttys003] [ut_user sledwards] [ut_id s003] [ut_line ttys003] [ut_pi
d 21931] [ut_type 8] [ut_tv.tv_sec 1363551169] [ut_tv.tv_usec 647288] [ASLExpireTime 1395173569]
[ASLMessageID 28599] [Time 2013-03-23 00:10:53Z] [TimeNanoSec 859756000] [Level 5] [PID 28503] [
UID 0] [GID 20] [ReadGID 80] [Host nibble.blah] [Sender login] [Facility com.apple.system.lastlo
g] [Message USER_PROCESS: 28503 ttys003] [ut_user sledwards] [ut_id s003] [ut_line ttys003] [ut_
pid 28503] [ut_type 7] [ut_tv.tv_sec 1363997453] [ut_tv.tv_usec 859054] [ASLExpireTime 139561985
3]
```

Major Log Changes in 10.12 and iOS 10: Unified Logging

Unified Logging: macOS 10.12, iOS 10.0, tvOS 10.0, watchOS 3.0

Supersedes Apple System Logs (ASL) and Syslog

macOS and iOS (Physical) Locations:

- /var/db/diagnostics/
- Referenced Data in: /var/db/uuidtext/

Binary Files: *.tracev3

```
Sarahs-MacBook-Pro:diagnostics oompa$ sw_vers
ProductName:    Mac OS X
ProductVersion: 10.13.1
BuildVersion:   17B1003
Sarahs-MacBook-Pro:diagnostics oompa$ pwd
/var/db/diagnostics
Sarahs-MacBook-Pro:diagnostics oompa$ ls -la
total 2848
drwxr-x---   10 root  admin      320 Dec 15 21:30 .
drwxr-xr-x   85 root  wheel     2720 Dec 16 03:46 ..
drwxr-xr-x    2 root  admin       64 Nov 11 13:17 HighVolume
drwxr-xr-x   55 root  admin     1760 Dec 16 13:19 Persist
drwxr-xr-x  407 root  admin    13024 Dec 16 17:09 Special
-rw-r--r--    1 root  admin   265220 Dec 16 17:11 logdata.statistics.0.txt
-rw-r--r--    1 root  admin  1049501 Dec 12 19:58 logdata.statistics.1.txt
-rw-r--r--    1 root  admin   125296 Dec 15 13:23 shutdown.log
drwxr-xr-x    3 root  wheel       96 Nov 11 13:19 timesync
-rw-r--r--    1 root  admin      484 Dec 15 21:30 version.plist
Sarahs-MacBook-Pro:diagnostics oompa$ ls -l Persist/
total 1058088
-rw-r--r--@ 1 root  admin  10489040 Dec  4 19:25 0000000000000095.tracev3
-rw-r--r--@ 1 root  admin  10483632 Dec  4 19:46 0000000000000096.tracev3
-rw-r--r--@ 1 root  admin  10483608 Dec  4 20:08 0000000000000097.tracev3
-rw-r--r--@ 1 root  admin  10487128 Dec  4 20:33 0000000000000098.tracev3
-rw-r--r--@ 1 root  admin  10482456 Dec  4 20:56 0000000000000099.tracev3
-rw-r--r--@ 1 root  admin  10490336 Dec  4 21:17 000000000000009a.tracev3
-rw-r--r--@ 1 root  admin  10484864 Dec  4 21:40 000000000000009b.tracev3
-rw-r--r--@ 1 root  admin  10482976 Dec  4 22:02 000000000000009c.tracev3
```

SANS | DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 110

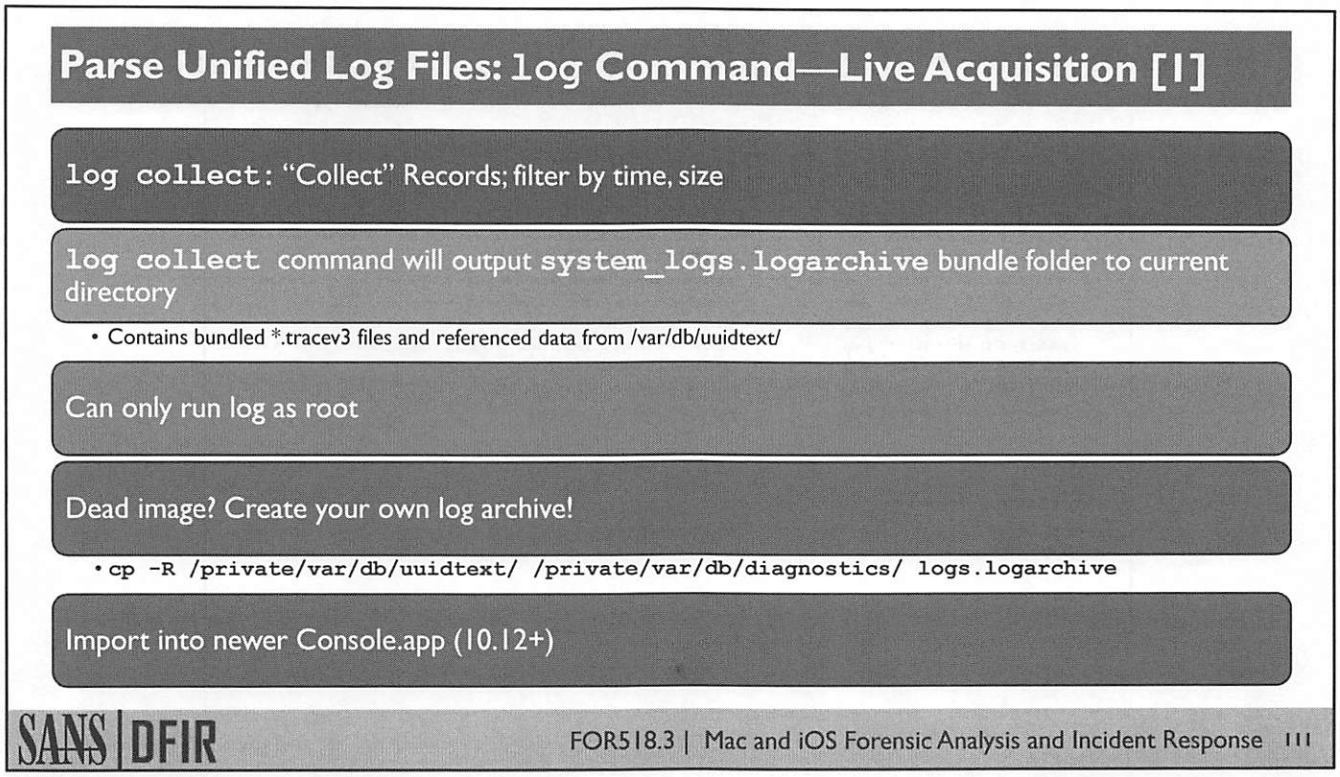
Introduced with macOS 10.12 and iOS 10 is unified logging, which is supposed to be uniform across devices and operating systems. This directory has changed over many point versions since it was released in 10.12. The screenshot above is from a 10.13.1 system, while it used to appear like the screenshot below in 10.12.

These log files are stored in the `/private/var/db/diagnostics/` directory and reference other data in the `/private/var/db/uuidtext/` directory. The Unified Logs are supposed to supersede ASL files, however ASL files still exist on the system and may still be worth investigating for certain items.

The main log files are stored in a binary format.

References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`

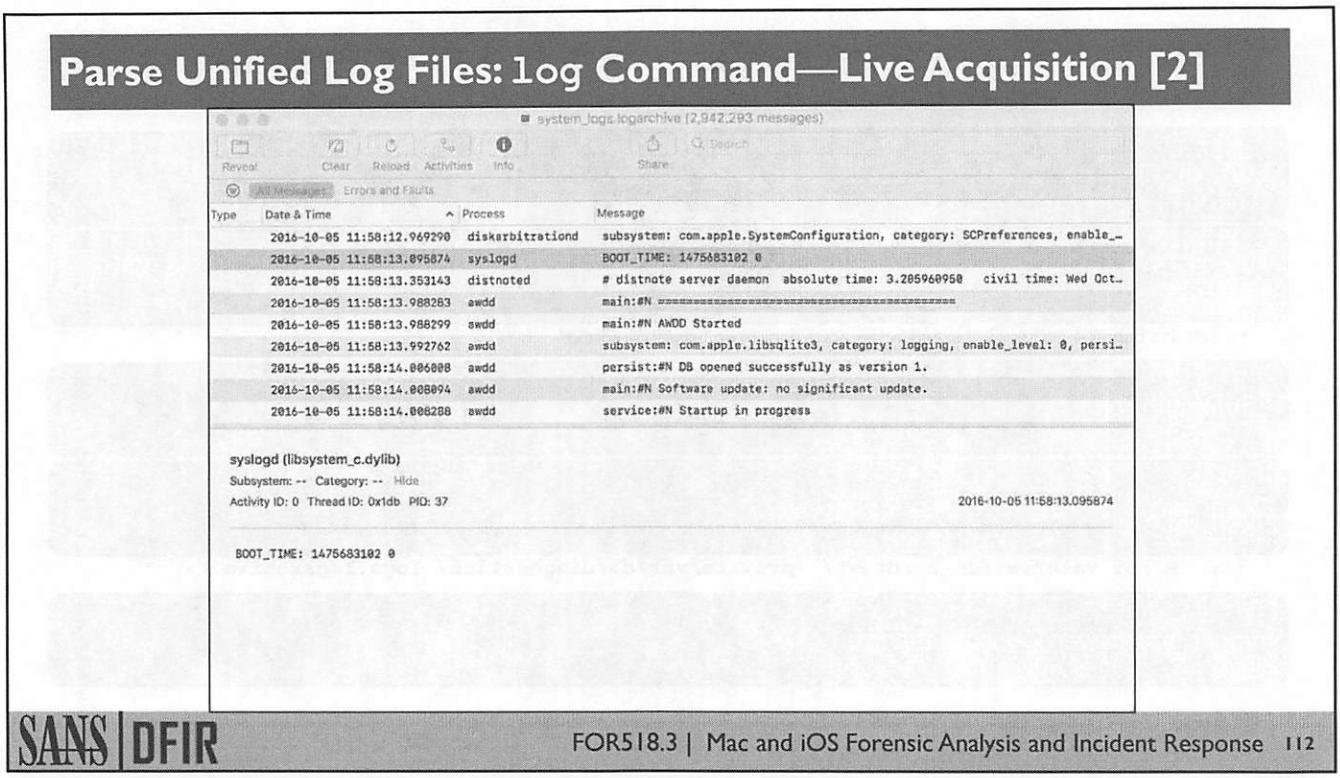
```
bash-3.2# pwd
/private/var/db/diagnostics
bash-3.2# ls -l
total 206272
drwxr-xr-x   2 root  wheel        68 Sep 27 19:03 Events
drwxr-xr-x  30 root  wheel      1020 Oct 22 21:52 FaultsAndErrors
drwxr-xr-x   2 root  wheel        68 Sep 27 19:03 Oversize
drwxr-xr-x   2 root  wheel        68 Sep 27 19:03 SpecialHandling
drwxr-xr-x   2 root  wheel        68 Sep 27 19:03 StateDumps
drwxr-xr-x  14 root  wheel       476 Oct 22 22:03 TTL
-rw-r-----   1 root  wheel   3428016 Oct 14 20:27 logdata.Persistent.20161014T005422.tracev3
-rw-r-----   1 root  wheel   9167544 Oct 15 13:11 logdata.Persistent.20161015T002740.tracev3
-rw-r-----   1 root  wheel  10491576 Oct 15 21:18 logdata.Persistent.20161015T171217.tracev3
-rw-r-----   1 root  wheel   3818584 Oct 17 19:22 logdata.Persistent.20161016T012225.tracev3
-rw-r-----   1 root  wheel   5675360 Oct 17 20:38 logdata.Persistent.20161017T232211.tracev3
-rw-r-----   1 root  wheel   6168968 Oct 18 20:20 logdata.Persistent.20161018T004411.tracev3
-rw-r-----   1 root  wheel   8171240 Oct 19 21:02 logdata.Persistent.20161019T004805.tracev3
-rw-r-----   1 root  wheel   9050872 Oct 21 19:38 logdata.Persistent.20161020T010211.tracev3
-rw-r-----   1 root  wheel  10497776 Oct 22 15:05 logdata.Persistent.20161022T000721.tracev3
-rw-r-----   1 root  wheel   5583528 Oct 22 21:52 logdata.Persistent.20161022T190750.tracev3
-rw-r-----   1 root  wheel  10537456 Oct 23 12:08 logdata.Persistent.20161023T015231.tracev3
-rw-r-----   1 root  wheel  10493240 Oct 23 13:21 logdata.Persistent.20161023T161202.tracev3
-rw-r-----   1 root  wheel  10567792 Oct 23 16:41 logdata.Persistent.20161023T172230.tracev3
-rw-r-----   1 root  wheel    903232 Oct 23 17:19 logdata.Persistent.20161023T204633.tracev3
-rw-r-----   1 root  wheel    718216 Oct 23 16:46 logdata.statistics.0.txt
-rw-r--r--   1 root  wheel    304313 Oct 15 13:11 shutdown.log
```



One way to acquire these files on a macOS is using the live method of collection using the "`log collect`" command. This command will bundle the files up into a bundle archive for later viewing in the Console.app application.

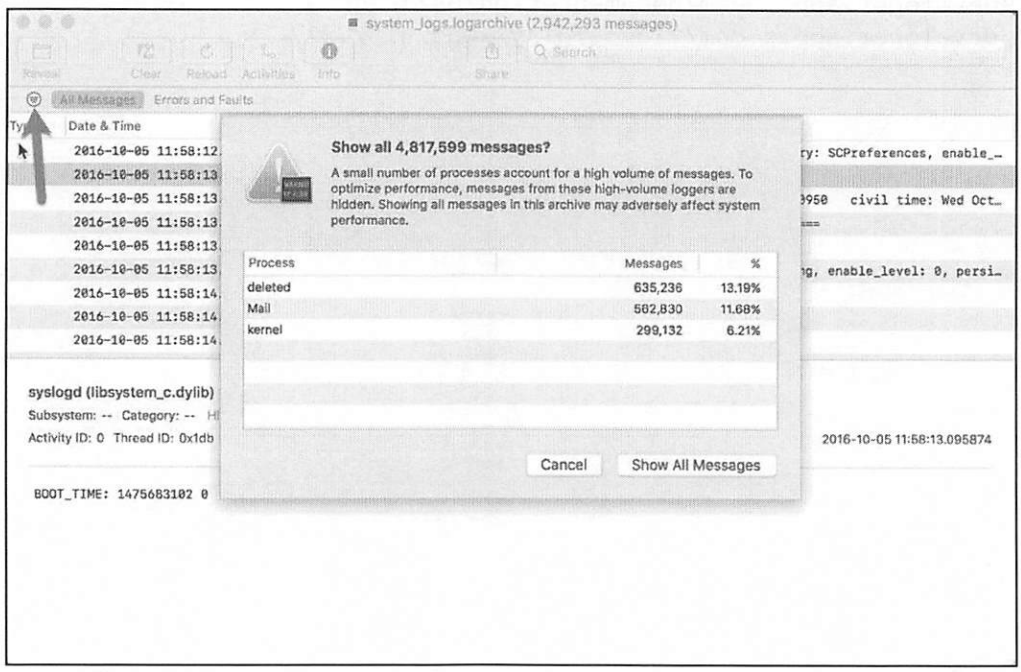
This command can only be run with root privileges.

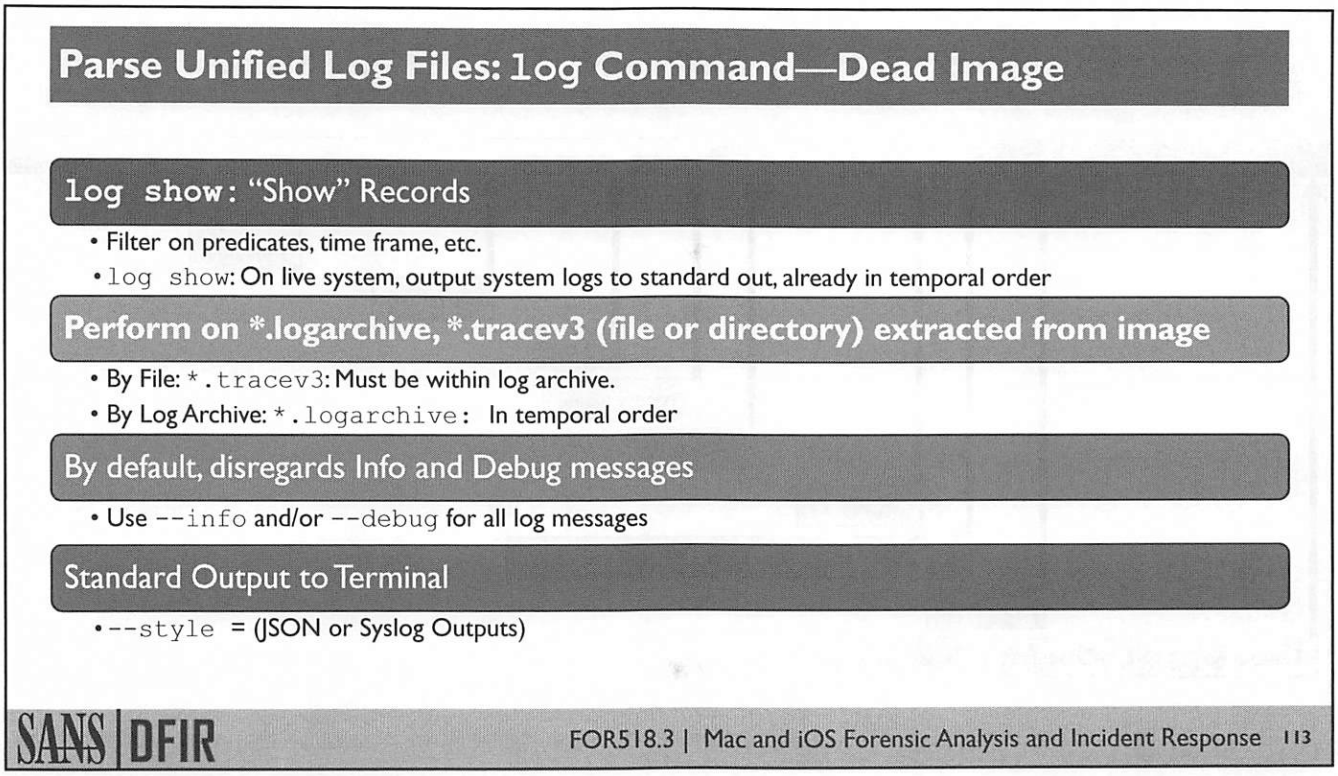
References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`



The new Console.app is shown above. The analyst can import the bundled log archive and review its contents. It is worth noting that not all messages are shown by default. Click the small button by "`All Messages`", as shown below, to view all messages.

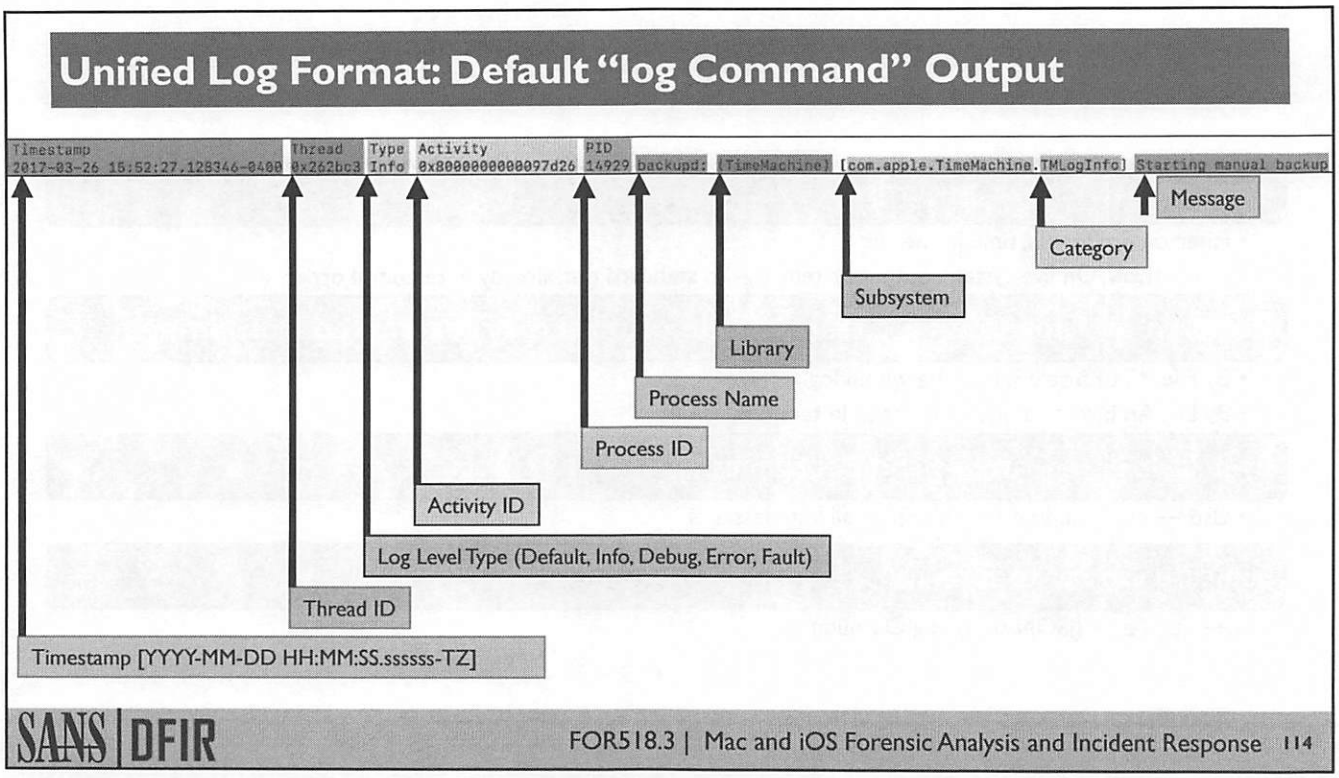
References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`





For the analyst working with a dead image, they can use the "`log show`" command to compile the `*.tracev3` files (or a log archive as needed). By default, this command will not include Info or Debug messages—additional arguments to the command will need to be used to acquire these messages.

References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`



The default unified log format is shown above. Each piece of the log message can be used to filter and search upon, as shown in future slides.

References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`

Unified Logs: Predicate Filtering "`--predicate`"

- Filter log events using logical statements (AND/OR/NOT/</>/=/CONTAINS/BEGINSWITH/TRUE/FALSE, etc.)
  - **eventType**: Filters on Event Types (Log, Trace, Activity)
  - **eventMessage**: Filters on message text or Activity Name (if log/trace event)
  - **messageType**: Filters on Log Level (Default, Info, Debug, Error, Fault)
  - **processImagePath**: Filters on the process name of the event originator
  - **senderImagePath**: Filters on sender name of the event originator (Library, framework, kext, mach-o binary)
  - **subsystem**: Filters on subsystem (reverse DNS ID)
  - **category**: Filters on subsystem category

[cd] = Ignore case/diacritics

```
bit:LogsUnite oompaS log show system_logs.logarchive/ --info  --predicate 'processImagePath contains[cd] "backupd" and category contains[cd] "TMLogInfo" and eventMessage
 contains "Starting"' --start "2017-03-25 15:00:00"
==========
/Users/oompa/Dropbox (Personal)/LogsUnite/system_logs.logarchive
==========
Skipping debug messages, pass --debug to include.
Filtering the log data using "processImagePath CONTAINS[cd] "backupd" AND category CONTAINS[cd] "TMLogInfo" AND eventMessage CONTAINS "Starting""
Timestamp                       Thread     Type        Activity             PID
2017-03-25 15:09:52.159122-0400 0xbb011    Info        0x0                  9802   backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting automatic backup
2017-03-25 15:12:37.464005-0400 0xbb011    Info        0x0                  9802   backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting post-backup thinning
2017-03-25 16:21:36.193812-0400 0x187e14   Info        0x0                  12334  backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting automatic backup
2017-03-25 16:26:59.910274-0400 0x187e14   Info        0x0                  12334  backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting post-backup thinning
2017-03-25 17:21:32.309421-0400 0x21ab5c   Info        0x0                  12566  backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting automatic backup
2017-03-25 17:26:55.642558-0400 0x21ab5c   Info        0x0                  12566  backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting post-backup thinning

Log      - Default:          0, Info:               6, Debug:            0, Error:          0, Fault:          0
Activity - Create:           0, Transition:         0, Actions:          0
```

SANS | DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 115

The "`--predicate`" argument allows an analyst to filter on certain pieces of a log message.

References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`

Unified Logs: Potential for Memory Analysis of Logs

- Look in the "diagnosticd" process

```
bit:unzip_mem oompa$ strings mem | grep "Received UDP DNS Response"
-- Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 64943 100 bytesXTUM
-- Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 60357 253 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:0000:0000:0000:17B8:54424 --
-- Received UDP DNS Response (flags 8183) RCODE: NXDomain (3) RD RA ID: 49294 119 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:0000:0000:0000:17B8:50683 --
-- Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 54986 188 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:0000:0000:0000:17B8:59691 --
-- Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 35998 216 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:042C:7360:EB74:D3E5:57952 -qA
Received UDP DNS Response (flags 8180) RCODE: NoErr (0) A ID: 35998 91 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:042C:7360:EB74:D3E5:57952 --qA
Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 41363 121 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:042C:7360:EB74:D3E5:60173 -qA
Received UDP DNS Response (flags 8180) RCODE: NoErr (0) A ID: 37510 103 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:042C:7360:EB74:D3E5:59930 -qA
Received UDP DNS Response (flags 8180) RCODE: NoErr (0) A ID: 63978 138 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:042C:7360:EB74:D3E5:61959 -qA
-- Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 29870 141 bytes from 2A
-- Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 52261 138 bytes from 10.11.12.254:53 to 10.11.12.214:51175 --
-- Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 1141 124 bytes from 2601:0141:0302:1F30:020D:B9FF:FE34:1C00:53 to 2601:0141:0302:1F30:0000:0000:0000:17B8:63995 --
```

Message	Volatile	Subsystem
-- Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 11920 69 bytes from 2601:0141:…	1	com.apple.mDNS…
1 Questions	1	com.apple.mDNS…
0 d.dropbox.com. Addr	1	com.apple.mDNS…
3 Answers	1	com.apple.mDNS…
0 TTL 76 17 d.dropbox.com. CNAME d.v.dropbox.com.	1	com.apple.mDNS…
1 TTL 39 4 d.v.dropbox.com. Addr 108.160.172.193	1	com.apple.mDNS…
2 TTL 39 4 d.v.dropbox.com. Addr 108.160.172.225	1	com.apple.mDNS…

With Unified Logs, some messages are only stored in memory. The example above shows a "`Volatile`" column in the Console.app application, where the message is marked volatile or not (1 = Volatile, 0 = Non-volatile). Looking for the same messages on a "dead" system may not show you all the messages you are expecting, because they were stored in memory.

The Unified Logs uses the "`diagnosticd`" process. As shown above, this is the process space where the volatile messages could be found.

References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`

Unified Logs Example: Network Usage – 'configd'

```
log show --info --predicate 'senderImagePath contains[cd]
"IPConfiguration" and (eventMessage contains[cd] "SSID" or
eventMessage contains[cd] "Lease" or eventMessage contains[cd]
"network changed")'
```

```
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] Removing Stale Lease 172.19.131.157 Router 172.19.191.2
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: no SSID
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: status = 'network changed'
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP bridge0: status = 'network changed'
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:23:ea:7f:4d:91
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: status = 'network changed'
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID is now hhonors (was United_Wi-Fi)
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: No lease for hhonors
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] AUTOMATIC-V6 en0: status = 'network changed'
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:23:ea:7f:4d:91
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:23:ea:7f:4d:91
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:23:ea:7f:4d:91
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stationx
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stationx
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stationx
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:24:13:8:7f:e1
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stationx
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stationx
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: INIT lease = { start 0x1e8efa9a, t1 0x1e8f01a2, t2 0x1e8f06e8, expiration 0x1e8f08aa }
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: SELECT lease = { start 0x1e8efa9c, t1 0x1e8f01a4, t2 0x1e8f06ea, expiration 0x1e8f08ac }
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] Saved lease for 107.16.207.185
```


Using the predicate-based query language, we can find the network usage of this system by finding all messages that contain the sender "`IPConfiguration`" and whose log message contains "`Lease`" or "`network changed`".

References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`

Unified Logs Example: User Logins via Apple Watch Device Information

```
log show --info system_logs.logarchive/ --predicate
"subsystem = "com.apple.sharing" and category =
"AutoUnlock" and eventMessage contains[cd] "Found Peer""
```

```
bit:LogsUnite oompa$ log show --info system_logs.logarchive/ --predicate 'subsystem = "com.apple.sharing" and category = "AutoUnlock" and eventMessage contai
ns[cd] "Found Peer"'
==========
/Users/oompa/Dropbox (Personal)/LogsUnite/system_logs.logarchive
==========
Skipping debug messages, pass --debug to include.
Filtering the log data using "subsystem == "com.apple.sharing" AND category == "AutoUnlock" AND eventMessage CONTAINS[cd] "Found Peer""
Timestamp                       Thread     Type        Activity             PID
2017-03-23 21:15:58.940782-0400 0x5e716    Default     0x0                  406    sharingd: [com.apple.sharing.AutoUnlock] Found peer callback
2017-03-23 21:15:58.940935-0400 0x5e716    Default     0x0                  406    sharingd: [com.apple.sharing.AutoUnlock] Found Peer:
        Device <name:miPhone7, uniqueID:F9B85FFC-2BC6-4E80-93DA-67508472C8F8, bluetooth ID:F768D25B-1EC8-476B-B4A3-D757890CD2E8, modelIdentifier:iPhone9,3>,
        Peer <SFBLEDevice ID f768d25b-1ec8-476b-b4a3-d757890cd2e8, AdvData '0a40', RSSI -47, 0, [-47], Name '?', Paired no>,
        Unlock Enabled: NO,
        Proxy Unlock Enabled: YES,
        Locked on Wrist: NO
2017-03-23 21:16:00.086856-0400 0x5e716    Default     0x0                  406    sharingd: [com.apple.sharing.AutoUnlock] Found peer callback
2017-03-23 21:16:00.087012-0400 0x5e716    Default     0x0                  406    sharingd: [com.apple.sharing.AutoUnlock] Found Peer:
        Device <name:miWatch, uniqueID:47B476E2-6B76-4D3D-B078-A24305AF1A21, bluetooth ID:C905A733-BEA0-4680-A41F-4DCDF577A099, modelIdentifier:Watch2,3>,
        Peer <SFBLEDevice ID c905a733-bea0-4680-a41f-4dcdf577a099, AdvData '0180', RSSI -38, 0, [-38], Name '?', Paired no>,
        Unlock Enabled: YES,
        Proxy Unlock Enabled: NO,
        Locked on Wrist: NO
```


This example shows the query that will produce logins that were performed with the Apple Watch. This one was performed on a `system_logs.logarchive` bundle.

This query is looking for a the "`com.apple.sharing`" subsystem, with the "`AutoUnlock`" category with the keywords "`Found Peer`" in the log message.

References:
Man Page for `log` Command
`https://developer.apple.com/documentation/os/logging`
`https://developer.apple.com/videos/play/wwdc2016/721/`

macOS Audit Logs: /private/var/audit/*

- Basic Security Module
- (BSM) Audit Logs
- Binary Format

```
sh-3.2# xxd 20130307232230.20130308000749
0000000: 1400 0000 7d0b af67 0000 5139 2136 0000  ....}..g..Q9!6..
0000010: 02ed 7101 0000 0000 0000 0000 0007 7366  ..q...........sf
0000020: 6c61 6773 002d 0200 0000 0000 0b61 6d5f  lags.-.......am_
0000030: 7375 6363 6573 7300 2d03 0000 0000 000b  success.-.......
0000040: 616d 5f66 6169 6c75 7265 0024 ffff ffff  am_failure.$....
0000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000060: 0000 0000 0001 8703 0000 0000 0000 0000  ................
0000070: 2700 0000 0000 13b1 0500 0000 7d14 0000  '...........}...
0000080: 007d 0baf 6800 0051 392b d400 0003 e771  .}..h..Q9+.....q
0000090: 0100 0000 0000 0000 0000 0773 666c 6167  ...........sflag
00000a0: 7300 2d02 0000 0000 000b 616d 5f73 7563  s.-.......am_suc
00000b0: 6365 7373 002d 0300 0000 0000 0b61 6d5f  cess.-.......am_
00000c0: 6661 696c 7572 6500 24ff ffff ff00 0000  failure.$.......
00000d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000e0: 0000 0187 0300 0000 0000 0000 0027 0000  .............'..
00000f0: 0000 0013 b105 0000 007d 1400 0000 7d0b  .........}....}.
0000100: af65 0000 5139 2bd5 0000 0000 7101 0000  .e..Q9+.....q...
0000110: 0000 0000 0000 0007 7366 6c61 6773 002d  ........sflags.-
0000120: 0200 0000 0000 0b61 6d5f 7375 6363 6573  .......am_succes
0000130: 7300 2d03 0000 0000 000b 616d 5f66 6169  s.-.......am_fai
0000140: 6c75 7265 0024 ffff ffff 0000 0000 0000  lure.$..........
0000150: 0000 0000 0000 0000 0000 0000 0000 0001  ................
0000160: 8705 0000 0000 0000 0000 2700 0000 0000  ..........'.....
0000170: 13b1 0500 0000 7d                        ......}
```


The audit logs are one of the few logs not located in the main log directories. These logs are located in the `/var/audit/` directory.

The audit logs use the Basic Security Module from OpenBSM (McAfee Research) based on the definitions created by Sun Microsystems. It has now been taken over by the TrustedBSD project.

The log data is stored in a binary format that can be viewed using tools native to OS X.

More information about BSM Audit logs can be found in Hal Pomeranz's article "Solaris Basic Security Mode (BSM) Auditing": `www.deer-run.com/~hal/sysadmin/SolarisBSMAuditing.html`.

```
sh-3.2# xxd 20130307232230.20130308000749
0000000: 1400 0000 7d0b af67 0000 5139 2136 0000  ....}..g..Q9!6..
0000010: 02ed 7101 0000 0000 0000 0000 0007 7366  ..q...........sf
0000020: 6c61 6773 002d 0200 0000 0000 0b61 6d5f  lags.-.......am_
0000030: 7375 6363 6573 7300 2d03 0000 0000 000b  success.-.......
0000040: 616d 5f66 6169 6c75 7265 0024 ffff ffff  am_failure.$....
0000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000060: 0000 0000 0001 8703 0000 0000 0000 0000  ................
0000070: 2700 0000 0000 13b1 0500 0000 7d14 0000  '...........}...
0000080: 007d 0baf 6800 0051 392b d400 0003 e771  .}..h..Q9+.....q
0000090: 0100 0000 0000 0000 0000 0773 666c 6167  ...........sflag
00000a0: 7300 2d02 0000 0000 000b 616d 5f73 7563  s.-.......am_suc
00000b0: 6365 7373 002d 0300 0000 0000 0b61 6d5f  cess.-.......am_
00000c0: 6661 696c 7572 6500 24ff ffff ff00 0000  failure.$.......
00000d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000e0: 0000 0187 0300 0000 0000 0000 0027 0000  .............'..
00000f0: 0000 0013 b105 0000 007d 1400 0000 7d0b  .........}....}.
0000100: af65 0000 5139 2bd5 0000 0000 7101 0000  .e..Q9+.....q...
0000110: 0000 0000 0000 0007 7366 6c61 6773 002d  ........sflags.-
0000120: 0200 0000 0000 0b61 6d5f 7375 6363 6573  .......am_succes
0000130: 7300 2d03 0000 0000 000b 616d 5f66 6169  s.-.......am_fai
0000140: 6c75 7265 0024 ffff ffff 0000 0000 0000  lure.$..........
0000150: 0000 0000 0000 0000 0000 0000 0000 0001  ................
0000160: 8705 0000 0000 0000 0000 2700 0000 0000  ..........'.....
0000170: 13b1 0500 0000 7d                        ......}
```

macOS Audit Logs: Audit Trail Files

- StartTime.EndTime
- YYYYMMDDHHMMSS.YYYYMMDDHHMMSS
- Other Filenames:
  - "current"
  - *.not_terminated
  - *.crash_recovery

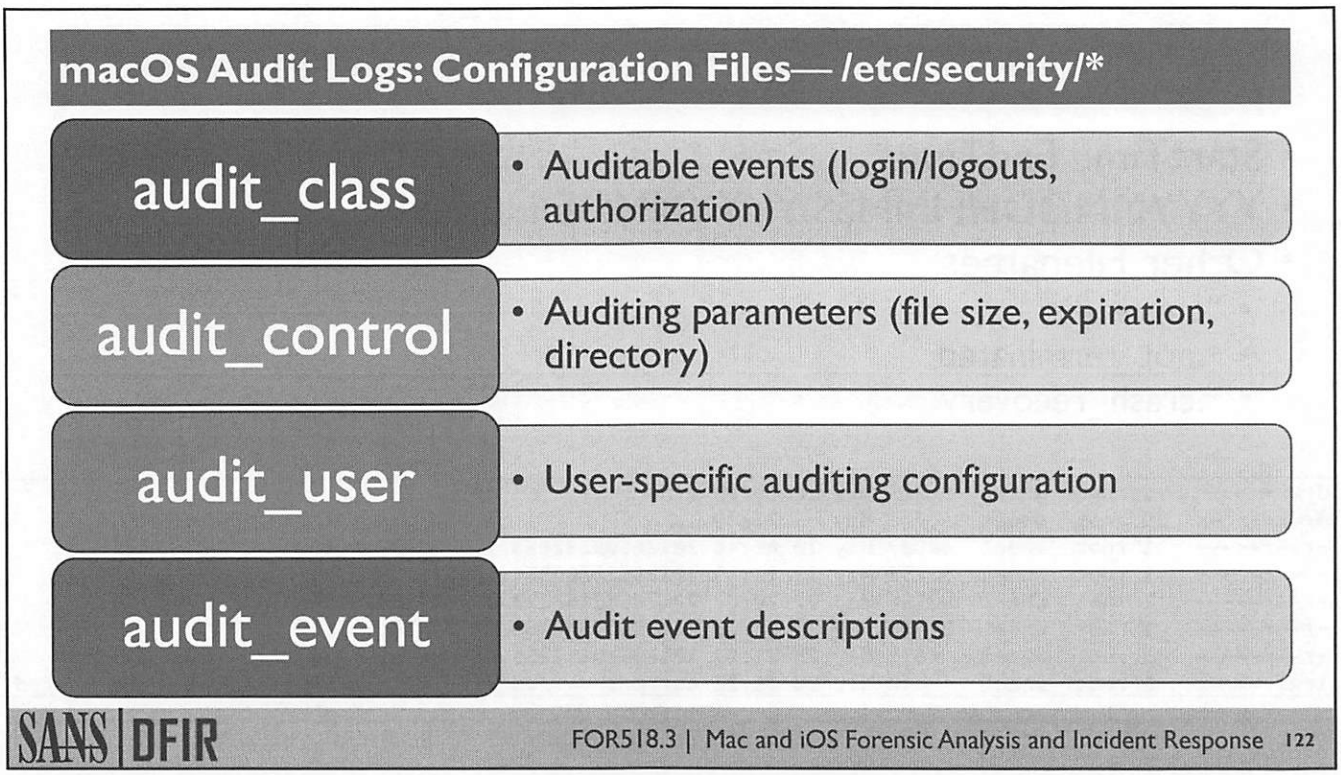
```
drwx------   8 root  wheel    272 May 28 15:22 .
drwxr-xr-x  29 root  wheel    986 May  9 21:39 ..
-r--r-----   1 root  wheel  48987 May 10 00:46 20120509232853.20120510044637
-r--r-----   1 root  wheel  57158 May 12 11:31 20120510204054.20120512153135
-r--r-----   1 root  wheel  92166 May 27 20:02 20120512153220.20120528000216
-r--r-----   1 root  wheel  20805 May 28 15:20 20120528000250.20120528192006
-r--r-----   1 root  wheel   4619 May 28 21:07 20120528192235.not_terminated
lrwxr-xr-x   1 root  wheel     40 May 28 15:22 current -> /var/audit/20120528192235.not_terminated
```


The log directory should contain many log files, each with a standard naming scheme (`StartTime.EndTime`) in the format `YYYYMMDDHHMMSS`. In the screenshot, the first log file (`20120509232853.20120510044637`) would contain data for the time period in between 05/09/2012 23:28:53 to 05/10/2012 04:64:37. Other filenames may also be used to show incompleteness or system error.

The "`current`" audit trail file will be a symbolic link to the active audit trail file, as shown in the screenshot above.

Those files ending with "`not_terminated`" are trail files that were not terminated properly due to a system error, or the file was otherwise inaccessible. The current audit trail that is in use will always have the "`not_terminated`" file extensions.

Audit trail files ending in "`crash_recovery`" are those files that were not terminated properly due to system or audit crash. The next audit trail file will have an "`audit crash recovery`" as the first record".



audit_class:
The `audit_class` file contains the auditable event class designations. Each of these will be used to determine which events are audited for the system or for a specific user.

audit_control:
The `audit_control` file contains the configuration data for the audit process.

This file contains different parameters.

- **`dir`**: Directory where audit trail files are stored.
- **`flags`**: Specifies audit event classes for a user (see audit_class file). `lo` = login/logout events, `aa` = audit administrative events.
- **`minfree`**: Minimum space (percentage) required on volume where audit logs are contained.
- **`naflags`**: Specifies audit event classes that are not attributable to a user (see audit_class file). "`naflags`" = non-attributable flags.
- **`policy`**: Global audit policy flags.
- **`filesz`**: Maximum audit trail file size (2 megabytes).
- **`expire-after`**: Expire audit trail files after a certain amount of time or size (expire after 10 megabytes of audit trail files).

audit_user:
Each user can have specific audit functions recorded. The default configuration only includes one user—root.

The format for each line is `username:alwaysaudit:neveraudit`, where the first parameter is the username, the second are those classes that should be audited, and the third are those classes that should not be audited.

audit_event:
The `audit_event` file contains auditable event types that are classified into various event classes. The entries in this file follow the format `eventnum:eventname:description:eventclass`.

- **`eventnum`**: Unique number for the event
- **`eventname:`** Event Name
- **`description`**: Event Description
- **`eventclass`**: Event class (see `audit_class`)

References:
`audit_class` Man Page
`audit_control` Man Page
`audit_user` Man Page
`audit_event` Man Page

praudit -xn /var/audit/* - su Example:

```
<record version="11" event="user authentication" modifier="0" time="Mon
May 28 21:12:51 2012" msec=" + 41 msec" >
<subject audit-uid="501" uid="0" gid="20" ruid="501" rgid="20" pid="552"
sid="100004" tid="552 0.0.0.0" />
<text>Verify password for record type Users &apos;root&apos; node
&apos;/Local/Default&apos;</text>
<return errval="success" retval="0" />
</record>

<record version="11" event="user authentication" modifier="0" time="Mon
May 28 21:12:55 2012" msec=" + 449 msec" >
<subject audit-uid="501" uid="0" gid="20" ruid="501" rgid="20" pid="554"
sid="100004" tid="554 0.0.0.0" />
<text>Verify password for record type Users &apos;root&apos; node
&apos;/Local/Default&apos;</text>
<return errval="failure: Unknown error: 255" retval="5000" />
</record>
```


The `praudit` (i.e., print audit) command can be used to view the audit log files. This command is native to OS X.

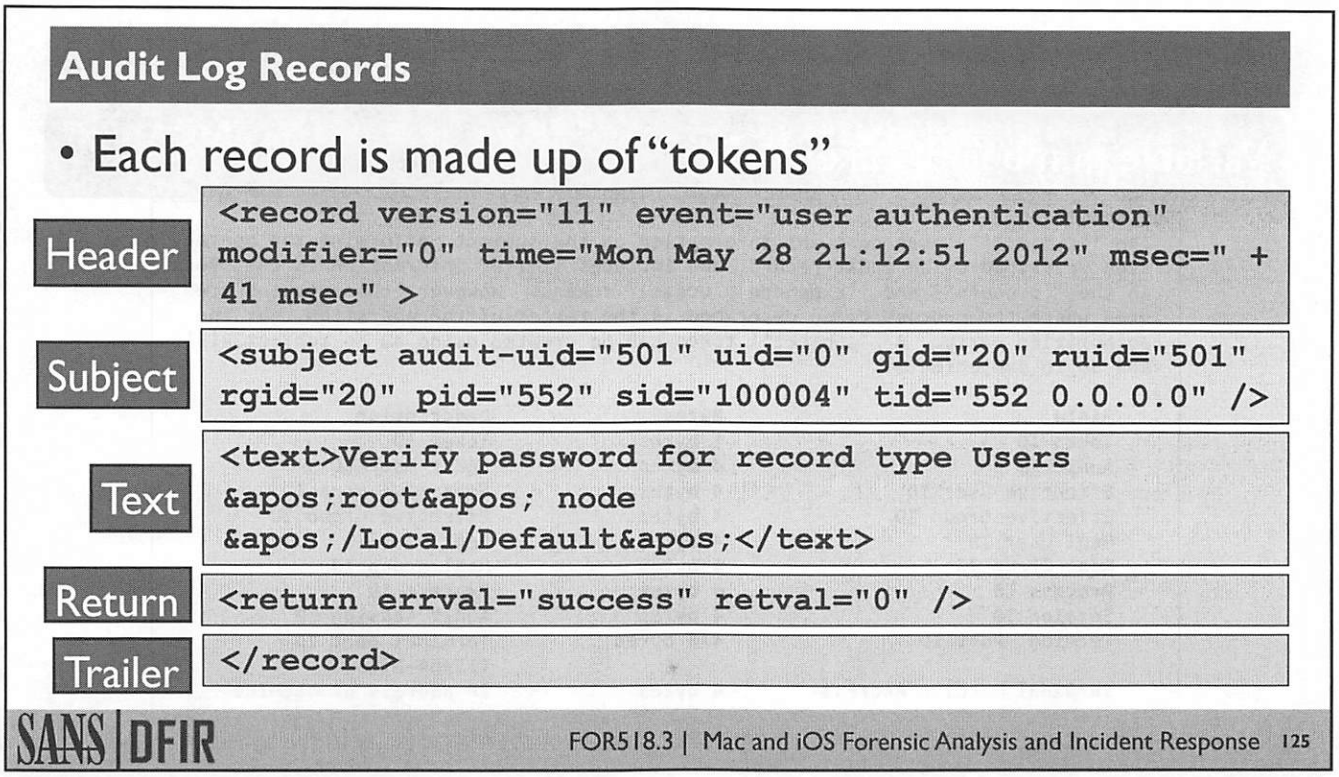
The `praudit` command shown above uses the `-xn` options to print the audit records in XML format (`-x`) and does not convert the user and group IDs (`-n`).

Shown above are two separate records. The first record contains the data consistent with a successful `su` logon, while the second contains a failed `su` logon.

The event identifier "`user authentication`" can be used to determine logon activities of users.

In the first record, the highlighted "`ruid`" key contains the UID for user 501 (usually the first user account created on the system). This user is attempting to use the "`su`" command to get root privileges, as shown in the "`text`" key. The "`return`" key shows that it was successful.

In the second record, the same event occurred but returned with a "`failure:Unknown error: 255`" error. This error type is returned at a failed "`su`" login.



As shown previously, each audit record is made up of various components, or "`tokens`".

The example above contains five of the basic tokens needed for each record. Each record may contain different tokens, depending on the contents of the data.

- **`Header`**: Required token. This is used to mark the start of a record. It contains data such as the record version, event type/modifier, timestamp, and record length. The length is not shown above due to how the record was printed.
- **`Subject`**: This token contains data associated with the "subject" making the operation. This record will have the audit user ID, effective user ID, effective group ID, real user ID, real group ID, process ID, session ID, and terminal ID. The real user IDs are generally the user executing the process (UID 501), while the effective user ID is the user the process is running under (i.e., root: UID 0).
- **`Text`**: The Text token contains a string with a description of the event.
- **`Return`**: The Return token contains a return value that may be used by the system.
- **`Trailer`**: Required token. This contains the record termination and may also contain a magic number or byte count, depending on how the record is printed.

Audit Log Record: Tokens

Variable number of tokens

Subject Token
The ``subject'' token contains information on the subject performing the operation described by an audit record, and includes similar information to that found in the ``process'' and ``expanded process'' tokens. However, those tokens are used where the process being described is the target of the operation, not the authorizing party. A ``subject'' token can be created using au_to_subject32(3) and au_to_subject64(3).

Field	Bytes	Description
Token ID	1 byte	Token ID
Audit ID	4 bytes	Audit user ID
Effective User ID	4 bytes	Effective user ID
Effective Group ID	4 bytes	Effective group ID
Real User ID	4 bytes	Real user ID
Real Group ID	4 bytes	Real group ID
Process ID	4 bytes	Process ID
Session ID	4 bytes	Audit session ID
Terminal Port ID	4/8 bytes	Terminal port ID (32/64-bits)
Terminal Machine Address	4 bytes	IP address of machine

More information about each token can be found on the `audit.log` man page.

Example from the man page for the `subject` token.

Reference:
`audit.log` Man Page

auditreduce Command

Filter audit records given:

- Before or after a date/time
- A specific user
- A specific subject token
- An audit event (shown below)

```
bash-3.2# auditreduce -m AUE_lw_login 20140716023538.20140725213130 | praudit -xn
<?xml version='1.0' encoding='UTF-8'?>
<audit>
<record version="11" event="loginwindow login" modifier="0" time="Tue Jul 15 22:47:21 2014" msec=" + 473 msec" >
<subject audit-uid="501" uid="0" gid="0" ruid="501" rgid="20" pid="73" sid="100004" tid="503316500.0.0.0" />
<return errval="success" retval="0" />
</record>
```


BSM audit records can be filtered using the `auditreduce` command.

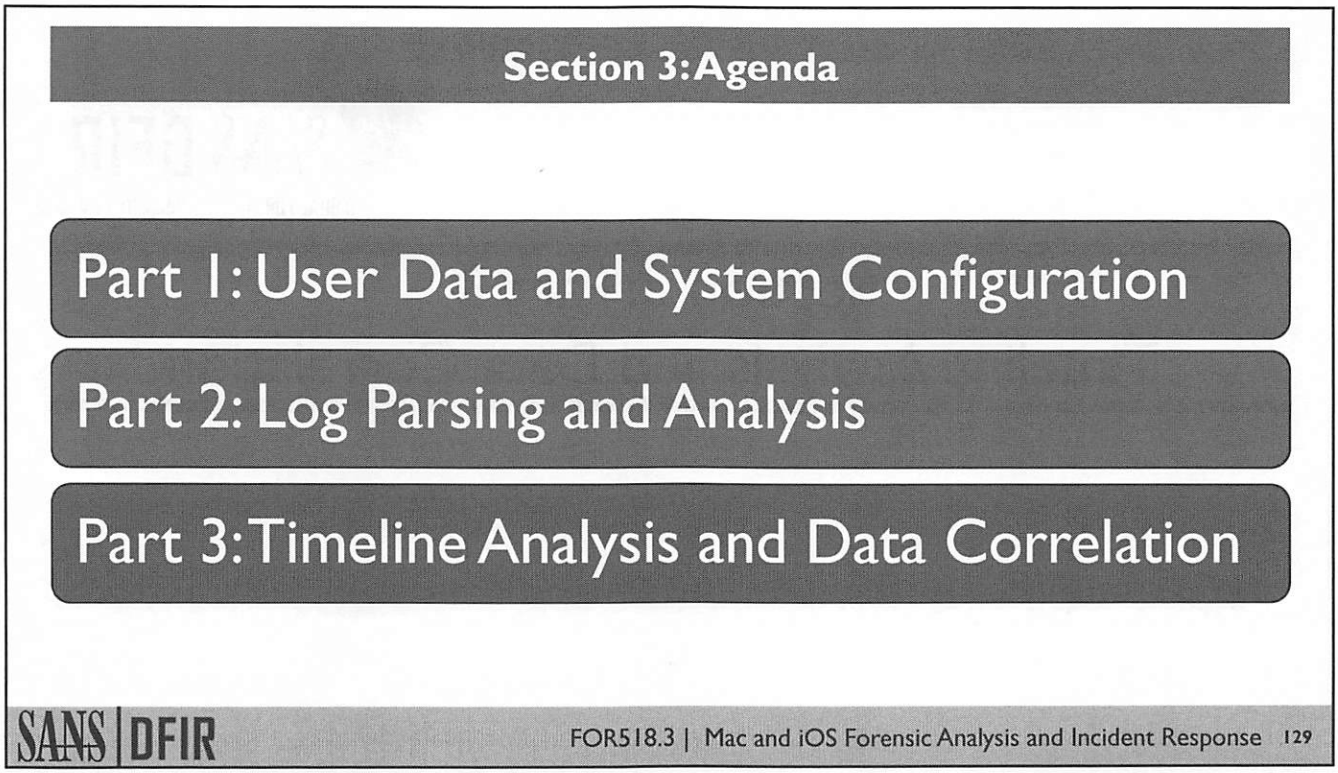
The screenshot shows a filter for a specific audit event to find login window logins (`AUE_lw_login`). The `auditreduce` command can be piped to the `praudit` command for easier viewing.

The audit events can be found in `/etc/security/audit_event`.

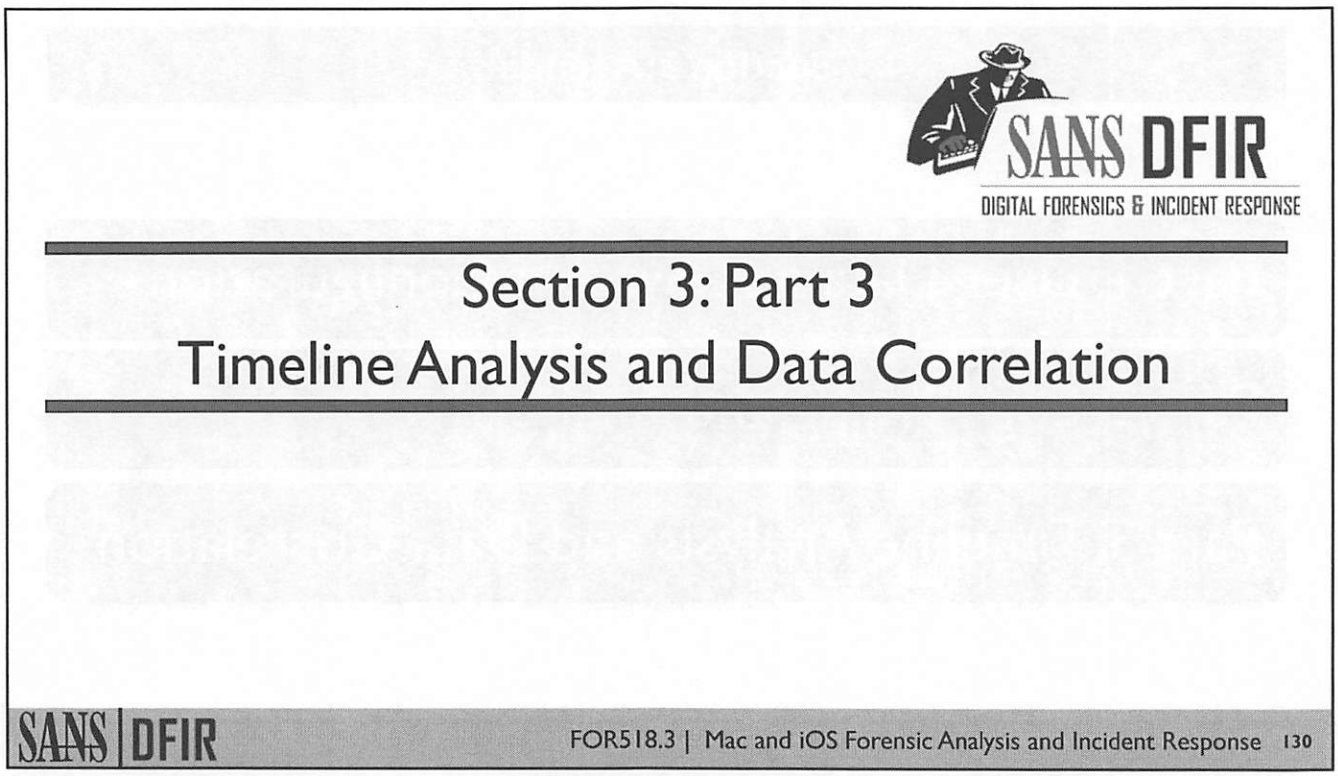
Reference:
`auditreduce` Man Page

Lab 3.3
Log Parsing and Analysis

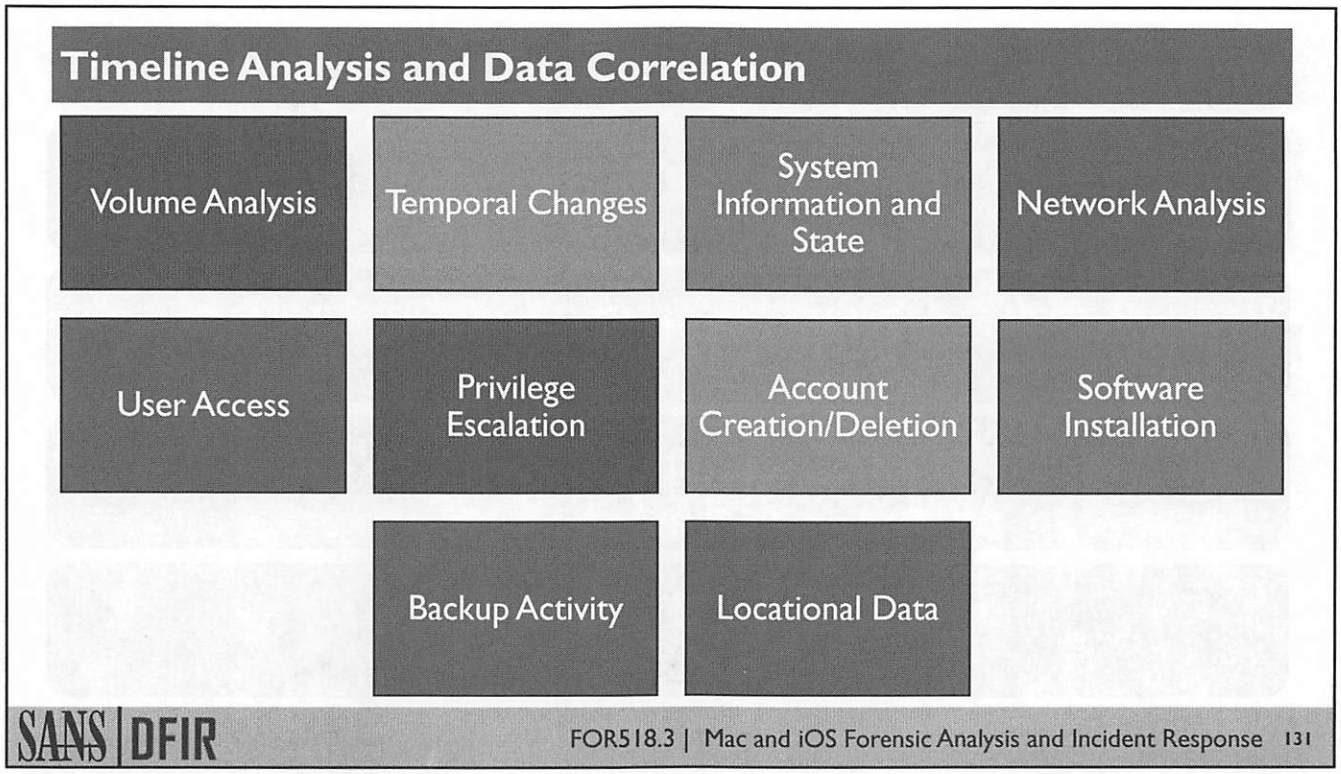
This page intentionally left blank.



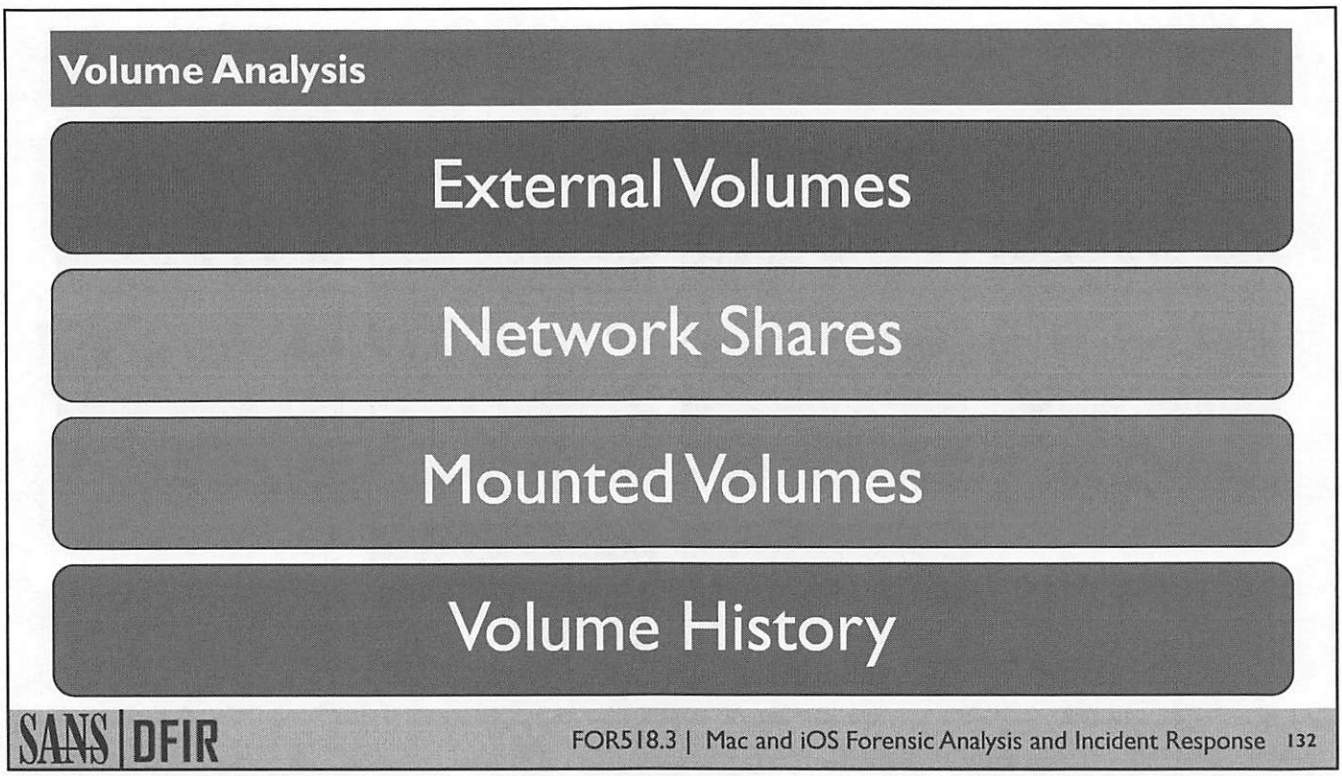
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.



Volume analysis can provide us with information about other items connected to the system or accessed via the system.

Search "/Volumes/": Unix Maintenance Log—daily.out

```
Sat Mar  3 11:10:50 EST 2018

Removing old temporary files:
  /tmp/powerlog

Cleaning out old system announcements:

Removing stale files from /var/rwho:

Disk status:
Filesystem     Size   Used  Avail Capacity iused               ifree %iused  Mounted on
/dev/disk3s1   30Gi   15Gi  5.4Gi    74%  602741 9223372036854173066    0%   /
/dev/disk5s1  7.2Gi   42Mi  7.1Gi     1%      89 9223372036854775718    0%   /Volumes/SEKRET

Network interface status:
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16384 <Link#1>                        31270     0    31270     0     0
lo0   16384 127           localhost          31270     -    31270     -     -
lo0   16384 localhd
lo0   16384 fe80::1
gif0* 1280  <Link#2
stf0* 1280  <Link#3
XHC20 0     <Link#4
en0   1500  <Link#5
en0   1500  davids-
en0   1500  192.168
en0   1500  fd82:58
en0   1500  fd82:58f6:b fd82:58f6:b61b::f   567345     -   569950     -     -
en1   1500  <Link#6>    72:00:00:b0:3b:60        0     0        0     0     0
en2   1500  <Link#7>    72:00:00:b0:3b:61        0     0        0     0     0
p2p0  2304  <Link#8>    0a:e8:56:37:ec:06        0     0        0     0     0
awdl0 1484  <Link#9>    3a:01:5d:65:e7:36    10592     0     4410     0     0
awdl0 1484  davids-macb fe80:9::3801:5dff    10592     -     4410     -     -
bridg 1500  <Link#10>   72:00:00:b0:3b:60        0     0        1     0     0
utun0 2000  <Link#11>                            0     0        2     0     0
utun0 2000  fe80::4c42: fe80:b::4c42:51e1        0     -        2     -     -

Local system status:
11:11  up 5 days, 17:47, 4 users, load averages: 3.97 1.92 1.59

-- End of daily output --
```

```
Sarahs-MBP:log oompa$ grep "/Volumes" daily.out
/dev/disk4s2   56Gi   11Gi   44Gi    21%  627728          4294339551    0%   /Volumes/TimeMachine
/dev/disk4s2   56Gi   11Gi   44Gi    21%  640056          4294327223    0%   /Volumes/TimeMachine
/dev/disk2s2   56Gi   11Gi   44Gi    21%  618580          4294348699    0%   /Volumes/TimeMachine
/dev/disk5s1  7.2Gi   42Mi  7.1Gi     1%      89 9223372036854775718    0%   /Volumes/SEKRET
```


Volume analysis may be important to an investigation to see what USB drives were mounted on the system, what software might have been installed, or what the historical usage of a certain volume is.

The term "`/Volumes/`" can be searched for in Unix periodic daily script output `daily.out` files. The `/Volumes/` directory is the default mount point for any volume that is mounted on the system.

The `daily.out` file shows what volumes were mounted on the system when the daily maintenance script was run. This log shows the device file the volume is using, the size of the volume, storage utilization, and the volume name and mount point.

There are three maintenance scripts that are run periodically. These scripts are located in `/etc/periodic` and are named `daily`, `weekly`, and `monthly`, after how often they are run. These scripts are started as launch daemons found in the `/System/Library/LaunchDaemons` directory.

- `com.apple.periodic-daily.plist`
- `com.apple.periodic-weekly.plist`
- `com.apple.periodic-monthly.plist`

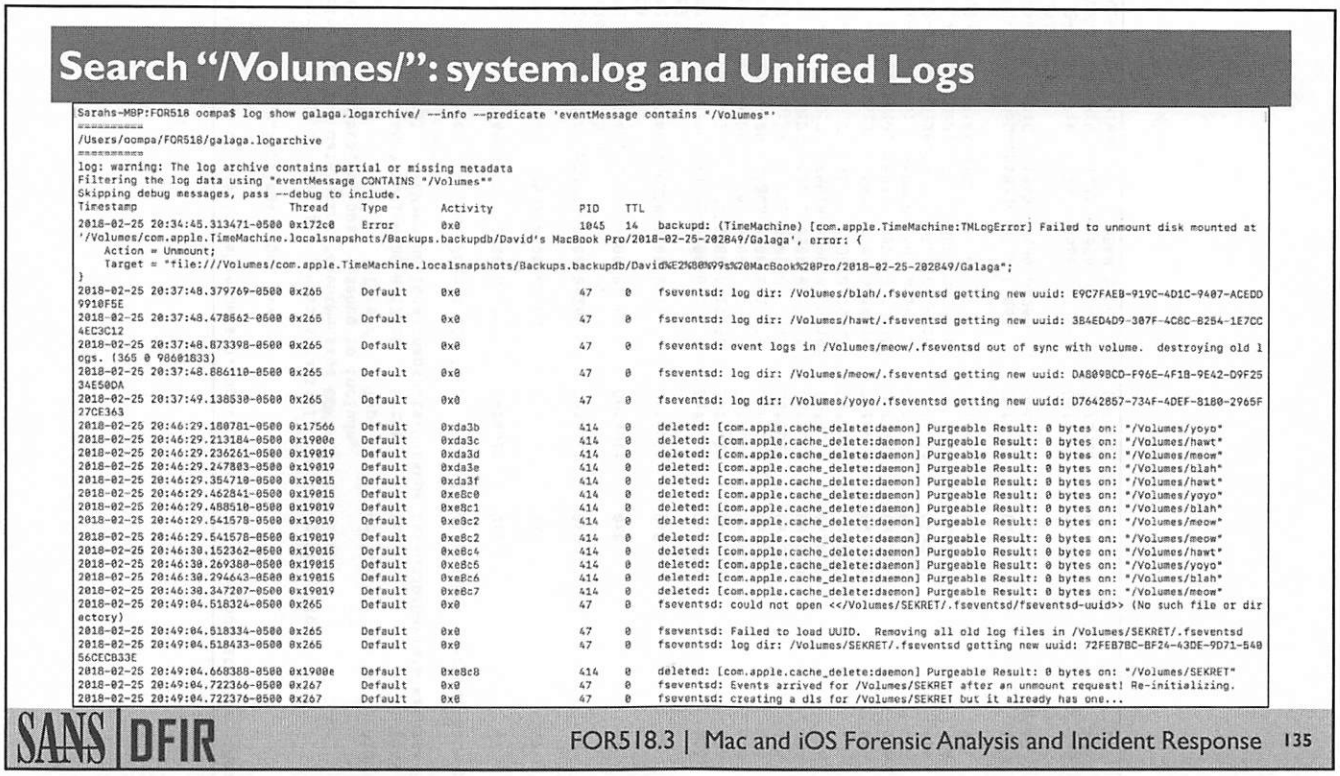
Each script produces a log file located in `/var/log/` named similar to the script that created it:

- **`daily.out`**: Contains output from temporary file removal, disk, network, and system status.
- **`weekly.out`**: Contains output from rebuilding the `whatis` database. This database contains a description of system commands.
- **`monthly.out`**: Contains output from rotating the fax logs and login accounting.

In older versions of OS X, these files may be named `daily.log`, `weekly.log`, and `monthly.log`.

References:
`periodic` Man Page
`periodic.conf` Man Page

```
Sat Mar  3 11:10:50 EST 2018

Removing old temporary files:
  /tmp/powerlog

Cleaning out old system announcements:

Removing stale files from /var/rwho:

Disk status:
Filesystem     Size   Used  Avail Capacity iused               ifree %iused  Mounted on
/dev/disk3s1   30Gi   15Gi  5.4Gi    74%  602741 9223372036854173066    0%   /
/dev/disk5s1  7.2Gi   42Mi  7.1Gi     1%      89 9223372036854775718    0%   /Volumes/SEKRET

Network interface status:
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16384 <Link#1>                        31270     0    31270     0     0
lo0   16384 127           localhost          31270     -    31270     -     -
lo0   16384 localhost   ::1                  31270     -    31270     -     -
lo0   16384 fe80::1%lo0 fe80:1::1            31270     -    31270     -     -
gif0* 1280  <Link#2>                            0     0        0     0     0
stf0* 1280  <Link#3>                            0     0        0     0     0
XHC20 0     <Link#4>                            0     0        0     0     0
en0   1500  <Link#5>    b8:e8:56:37:ec:06   567345     0   569950     0     0
en0   1500  davids-macb fe80:5::10a9:5f32   567345     -   569950     -     -
en0   1500  192.168.101   davids-mbp.lan    567345     -   569950     -     -
en0   1500  fd82:58f6:b fd82:58f6:b61b::2   567345     -   569950     -     -
en0   1500  fd82:58f6:b fd82:58f6:b61b::f   567345     -   569950     -     -
en1   1500  <Link#6>    72:00:00:b0:3b:60       0     0        0     0     0
en2   1500  <Link#7>    72:00:00:b0:3b:61       0     0        0     0     0
p2p0  2304  <Link#8>    0a:e8:56:37:ec:06       0     0        0     0     0
awdl0 1484  <Link#9>    3a:01:5d:65:e7:36   10592     0     4410     0     0
awdl0 1484  davids-macb fe80:9::3801:5dff   10592     -     4410     -     -
bridg 1500  <Link#10>   72:00:00:b0:3b:60       0     0        1     0     0
utun0 2000  <Link#11>                           0     0        2     0     0
utun0 2000  fe80::4c42: fe80:b::4c42:51e1       0     -        2     -     -

Local system status:
11:11  up 5 days, 17:47, 4 users, load averages: 3.97 1.92 1.59

-- End of daily output --
```

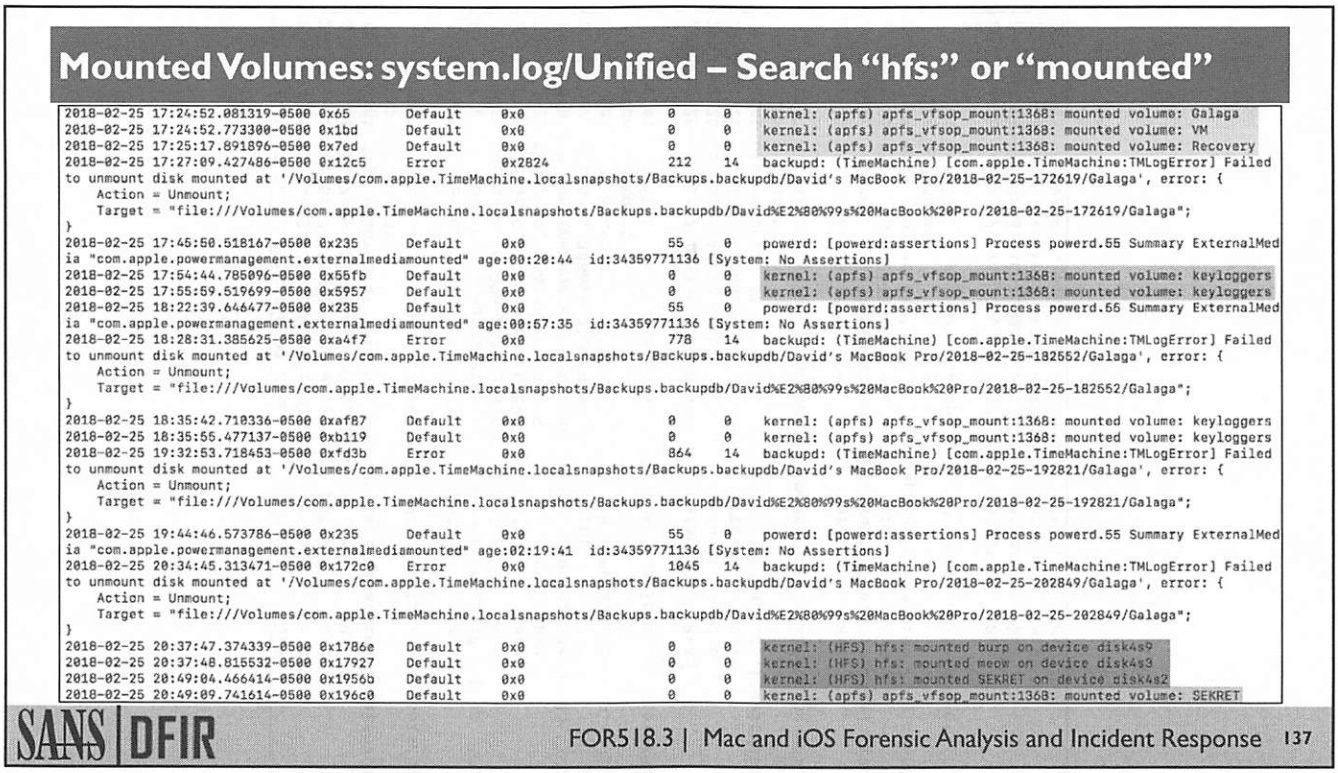


The term "`/Volumes/`" can be searched for in the `system.log` and `unified logs`.

These log files may show when the volume was automatically mounted and the UUID attached to that volume (`fseventsd`). It is important to point out that the same USB drive will have a different UUID every time it is mounted on the system.

Other processes may show other volume access, such as the "`backup`" process for Time Machine shown above.

```
Sarahs-MBP:FOR518 oompa$ log show galaga.logarchive/ --info --predicate 'eventMessage contains "/Volumes"'
==========
/Users/oompa/FOR518/galaga.logarchive
==========
log: warning: The log archive contains partial or missing metadata
Filtering the log data using "eventMessage CONTAINS "/Volumes""
Skipping debug messages, pass --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2018-02-25 20:34:45.313471-0500 0x172c0    Error       0x0                  1045   14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed to unmount disk mounted at
'/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-202849/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-202849/Galaga";
}
2018-02-25 20:37:48.379769-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/blah/.fseventsd getting new uuid: E9C7FAEB-919C-4D1C-9407-ACEDD
9910F5E
2018-02-25 20:37:48.478562-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/hawt/.fseventsd getting new uuid: 3B4ED4D9-307F-4C8C-8254-1E7CC
4EC3C12
2018-02-25 20:37:48.873398-0500 0x265      Default     0x0                  47     0    fseventsd: event logs in /Volumes/meow/.fseventsd out of sync with volume.  destroying old l
ogs. (365 0 98601833)
2018-02-25 20:37:48.886110-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/meow/.fseventsd getting new uuid: DA809BCD-F96E-4F1B-9E42-D9F25
34E50DA
2018-02-25 20:37:49.138530-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/yoyo/.fseventsd getting new uuid: D7642857-734F-4DEF-8180-2965F
27CE363
2018-02-25 20:46:29.180781-0500 0x17566    Default     0xda3b               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/yoyo"
2018-02-25 20:46:29.213184-0500 0x1900e    Default     0xda3c               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/hawt"
2018-02-25 20:46:29.236261-0500 0x19019    Default     0xda3d               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/meow"
2018-02-25 20:46:29.247803-0500 0x19019    Default     0xda3e               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/blah"
2018-02-25 20:46:29.354710-0500 0x19015    Default     0xda3f               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/hawt"
2018-02-25 20:46:29.462841-0500 0x19015    Default     0xe8c0               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/yoyo"
2018-02-25 20:46:29.488510-0500 0x19019    Default     0xe8c1               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/blah"
2018-02-25 20:46:29.541578-0500 0x19019    Default     0xe8c2               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/meow"
2018-02-25 20:46:29.541578-0500 0x19019    Default     0xe8c2               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/meow"
2018-02-25 20:46:30.152362-0500 0x19015    Default     0xe8c4               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/hawt"
2018-02-25 20:46:30.269380-0500 0x19015    Default     0xe8c5               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/yoyo"
2018-02-25 20:46:30.294643-0500 0x19015    Default     0xe8c6               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/blah"
2018-02-25 20:46:30.347207-0500 0x19019    Default     0xe8c7               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/meow"
2018-02-25 20:49:04.518324-0500 0x265      Default     0x0                  47     0    fseventsd: could not open <</Volumes/SEKRET/.fseventsd/fseventsd-uuid>> (No such file or dir
ectory)
2018-02-25 20:49:04.518334-0500 0x265      Default     0x0                  47     0    fseventsd: Failed to load UUID.  Removing all old log files in /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:04.518433-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/SEKRET/.fseventsd getting new uuid: 72FEB7BC-BF24-43DE-9D71-540
56CECB33E
2018-02-25 20:49:04.668388-0500 0x1900e    Default     0xe8c8               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:04.722366-0500 0x267      Default     0x0                  47     0    fseventsd: Events arrived for /Volumes/SEKRET after an unmount request! Re-initializing.
2018-02-25 20:49:04.722376-0500 0x267      Default     0x0                  47     0    fseventsd: creating a dls for /Volumes/SEKRET but it already has one...
```

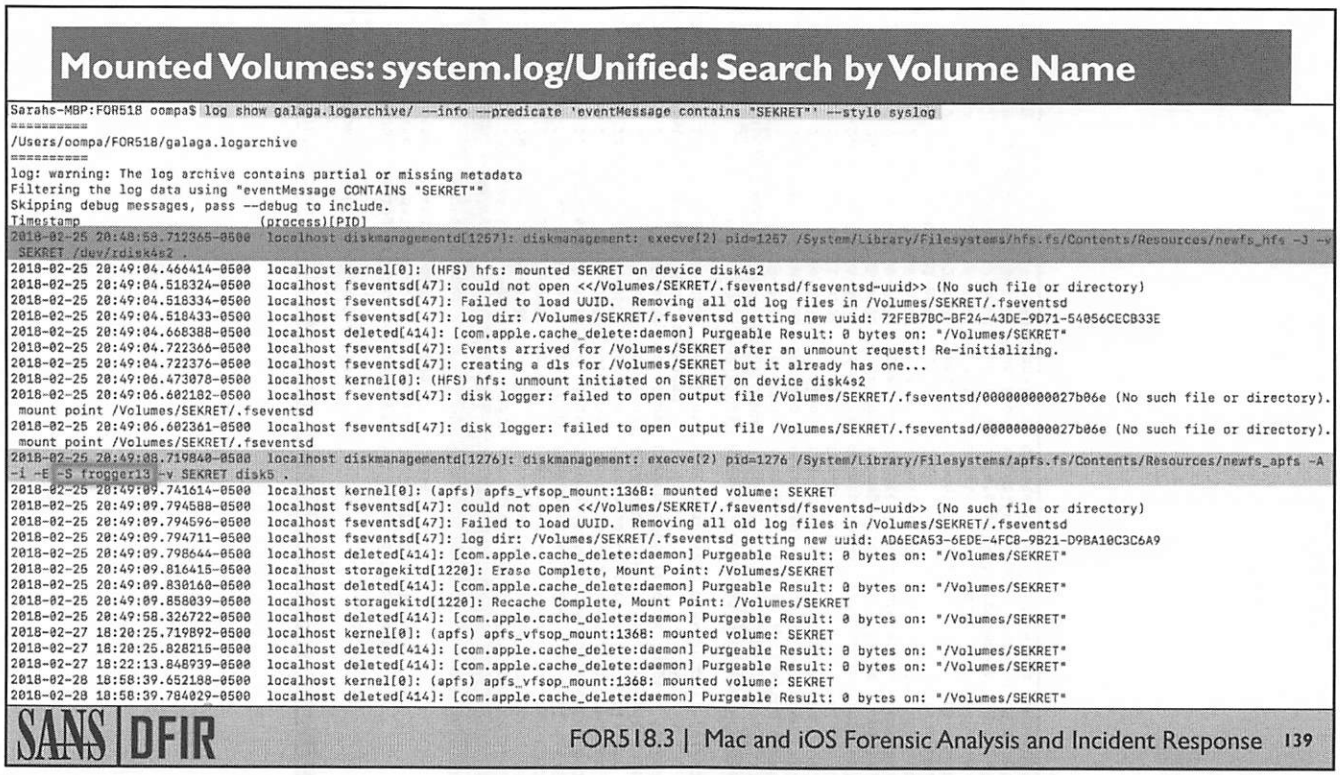


Older versions of this search will show when volumes were mounted and unmounted (shown below). Newer logs (above) do not specifically show when a volume was unmounted.

These records can be found in the `system.log` or the Unified logs by searching for "`hfs:`", "`mounted`", or "`unmounted`". Just because the keyword being used to search is "`hfs:`" does not mean that this limits the search to only HFS volumes—all mounted volumes will be shown.

```
Aug  2 09:00:28 nibble kernel[0]: hfs: mounted Recovery HD on device disk0s3
Aug  2 09:00:29 nibble kernel[0]: hfs: unmount initiated on Recovery HD on device disk0s3
Aug  2 09:00:51 nibble kernel[0]: hfs: mounted Recovery HD on device disk0s3
Aug  2 09:00:52 nibble kernel[0]: hfs: unmount initiated on Recovery HD on device disk0s3
Aug  2 09:30:13 nibble kernel[0]: hfs: mounted ExifTool-9.69 on device disk2
Aug  2 13:10:11 nibble kernel[0]: hfs: mounted Thunderbolt on device disk3s3
Aug  2 13:11:09 nibble kernel[0]: hfs: unmount initiated on Thunderbolt on device disk3s3
Aug  2 14:55:30 nibble kernel[0]: hfs: mounted Recovery HD on device disk0s3
Aug  2 14:55:30 nibble kernel[0]: hfs: unmount initiated on Recovery HD on device disk0s3
Aug  2 15:29:53 nibble kernel[0]: hfs: mounted Thunderbolt on device disk3s3
Aug  2 15:31:29 nibble kernel[0]: hfs: unmount initiated on Thunderbolt on device disk3s3
Aug  2 15:33:09 nibble kernel[0]: hfs: mounted Thunderbolt on device disk3s3
Aug  2 15:33:13 nibble kernel[0]: hfs: unmount initiated on Thunderbolt on device disk3s3
Aug  2 15:35:21 nibble kernel[0]: hfs: mounted Thunderbolt on device disk3s3
Aug  2 15:35:49 nibble kernel[0]: hfs: unmount initiated on Thunderbolt on device disk3s3
Aug  2 15:35:59 nibble kernel[0]: hfs: mounted Thunderbolt on device disk3s3
Aug  2 15:36:18 nibble kernel[0]: hfs: unmount initiated on Thunderbolt_External_Drive on device disk3s3
Aug  2 15:36:31 nibble kernel[0]: hfs: mounted Thunderbolt_External_Drive on device disk3s3
Aug  2 15:37:43 nibble kernel[0]: hfs: unmount initiated on Thunderbolt_External_Drive on device disk3s3
Aug  2 15:37:58 nibble kernel[0]: hfs: mounted Thunderbolt_External_Drive on device disk3s3
Aug  2 15:38:31 nibble kernel[0]: hfs: unmount initiated on Thunderbolt_External_Drive on device disk3s3
```

```
2018-02-25 17:24:52.081319-0500 0x65       Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: Galaga
2018-02-25 17:24:52.773300-0500 0x1bd      Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: VM
2018-02-25 17:25:17.891896-0500 0x7ed      Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: Recovery
2018-02-25 17:27:09.427486-0500 0x12c5     Error       0x2824               212    14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed
to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-172619/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-172619/Galaga";
}
2018-02-25 17:45:50.518167-0500 0x235      Default     0x0                  55     0    powerd: [powerd:assertions] Process powerd.55 Summary ExternalMed
ia "com.apple.powermanagement.externalmediamounted" age:00:20:44  id:34359771136 [System: No Assertions]
2018-02-25 17:54:44.785096-0500 0x55fb     Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: keyloggers
2018-02-25 17:55:59.519699-0500 0x5957     Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: keyloggers
2018-02-25 18:22:39.646477-0500 0x235      Default     0x0                  55     0    powerd: [powerd:assertions] Process powerd.55 Summary ExternalMed
ia "com.apple.powermanagement.externalmediamounted" age:00:57:35  id:34359771136 [System: No Assertions]
2018-02-25 18:28:31.385625-0500 0xa4f7     Error       0x0                  778    14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed
to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-182552/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-182552/Galaga";
}
2018-02-25 18:35:42.710336-0500 0xaf87     Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: keyloggers
2018-02-25 18:35:55.477137-0500 0xb119     Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: keyloggers
2018-02-25 19:32:53.718453-0500 0xfd3b     Error       0x0                  864    14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed
to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-192821/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-192821/Galaga";
}
2018-02-25 19:44:46.573786-0500 0x235      Default     0x0                  55     0    powerd: [powerd:assertions] Process powerd.55 Summary ExternalMed
ia "com.apple.powermanagement.externalmediamounted" age:02:19:41  id:34359771136 [System: No Assertions]
2018-02-25 20:34:45.313471-0500 0x172c0    Error       0x0                  1045   14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed
to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-202849/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-202849/Galaga";
}
2018-02-25 20:37:47.374339-0500 0x1786e    Default     0x0                  0      0    kernel: (HFS) hfs: mounted burp on device disk4s9
2018-02-25 20:37:48.815532-0500 0x17927    Default     0x0                  0      0    kernel: (HFS) hfs: mounted meow on device disk4s3
2018-02-25 20:49:04.466414-0500 0x1956b    Default     0x0                  0      0    kernel: (HFS) hfs: mounted SEKRET on device disk4s2
2018-02-25 20:49:09.741614-0500 0x196c0    Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
```



Once the name of a volume of interest has been identified, all logs can be searched for related activity to that volume.

The screenshot above shows the activity for a volume named "SEKRET". Some of the items you may find in addition to mounting activity and volume use is volume formatting—note the highlighted entries. These are the artifacts created of an encrypted APFS (from an HFS+ volume) volume named "SEKRET". On 10.13 and 10.13.1 systems, you may find the password to unlock the volume. In the example above, it is "frogger13".

```
Sarahs-MBP:FOR518 oompa$ log show galaga.logarchive/ --info --predicate 'eventMessage contains "SEKRET"' --style syslog
==========
/Users/oompa/FOR518/galaga.logarchive
==========
log: warning: The log archive contains partial or missing metadata
Filtering the log data using "eventMessage CONTAINS "SEKRET""
Skipping debug messages, pass --debug to include.
Timestamp                       (process)[PID]
2018-02-25 20:48:58.712365-0500  localhost diskmanagementd[1257]: diskmanagement: execve(2) pid=1257 /System/Library/Filesystems/hfs.fs/Contents/Resources/newfs_hfs -J -v
 SEKRET /dev/rdisk4s2 .
2018-02-25 20:49:04.466414-0500  localhost kernel[0]: (HFS) hfs: mounted SEKRET on device disk4s2
2018-02-25 20:49:04.518324-0500  localhost fseventsd[47]: could not open <</Volumes/SEKRET/.fseventsd/fseventsd-uuid>> (No such file or directory)
2018-02-25 20:49:04.518334-0500  localhost fseventsd[47]: Failed to load UUID.  Removing all old log files in /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:04.518433-0500  localhost fseventsd[47]: log dir: /Volumes/SEKRET/.fseventsd getting new uuid: 72FEB7BC-BF24-43DE-9D71-54056CECB33E
2018-02-25 20:49:04.668388-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:04.722366-0500  localhost fseventsd[47]: Events arrived for /Volumes/SEKRET after an unmount request! Re-initializing.
2018-02-25 20:49:04.722376-0500  localhost fseventsd[47]: creating a dls for /Volumes/SEKRET but it already has one...
2018-02-25 20:49:06.473078-0500  localhost kernel[0]: (HFS) hfs: unmount initiated on SEKRET on device disk4s2
2018-02-25 20:49:06.602182-0500  localhost fseventsd[47]: disk logger: failed to open output file /Volumes/SEKRET/.fseventsd/000000000027b06e (No such file or directory).
 mount point /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:06.602361-0500  localhost fseventsd[47]: disk logger: failed to open output file /Volumes/SEKRET/.fseventsd/000000000027b06e (No such file or directory).
 mount point /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:08.719840-0500  localhost diskmanagementd[1276]: diskmanagement: execve(2) pid=1276 /System/Library/Filesystems/apfs.fs/Contents/Resources/newfs_apfs -A
-i -E -S frogger13 -v SEKRET disk5 .
2018-02-25 20:49:09.741614-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-25 20:49:09.794588-0500  localhost fseventsd[47]: could not open <</Volumes/SEKRET/.fseventsd/fseventsd-uuid>> (No such file or directory)
2018-02-25 20:49:09.794596-0500  localhost fseventsd[47]: Failed to load UUID.  Removing all old log files in /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:09.794711-0500  localhost fseventsd[47]: log dir: /Volumes/SEKRET/.fseventsd getting new uuid: AD6ECA53-6EDE-4FC8-9B21-D9BA10C3C6A9
2018-02-25 20:49:09.798644-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:09.816415-0500  localhost storagekitd[1220]: Erase Complete, Mount Point: /Volumes/SEKRET
2018-02-25 20:49:09.830160-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:09.858039-0500  localhost storagekitd[1220]: Recache Complete, Mount Point: /Volumes/SEKRET
2018-02-25 20:49:58.326722-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-27 18:20:25.719892-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-27 18:20:25.828215-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-27 18:22:13.848939-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-28 18:58:39.652188-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-28 18:58:39.784029-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
```

USB Drives: Search "USBMSC"—system.log and Unified

Serial Number, Vendor ID, Product ID, Version

- 10.12+: Unified Logs
- 10.8–10.11: system.log
- **`USBMSC Identifier (non-unique): FBF1011220504638 0x90c 0x1000 0x1100`**

```
Sarahs-MBP:FOR518 oompa$ log show galaga.logarchive/ --info --predicate 'eventMessage contains "USBMSC"'
==========
/Users/oompa/FOR518/galaga.logarchive
==========
log: warning: The log archive contains partial or missing metadata
Filtering the log data using "eventMessage CONTAINS "USBMSC""
Skipping debug messages, pass --debug to include.
Timestamp                       Thread     Type        Activity            PID    TTL
2018-02-07 03:49:50.586839-0500 0xbe       Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): AA01102412155309367 8 0x781 0x5580 0x10, 3
2018-02-07 03:49:50.815848-0500 0xbe       Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 04:39:37.971787-0500 0x46c3     Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 05:23:44.240661-0500 0x7e2a     Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 06:14:59.238835-0500 0xab1d     Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 06:57:30.434931-0500 0xe060     Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 08:05:12.588033-0500 0x15ae9    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 09:20:47.558759-0500 0x16705    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 14:37:00.136479+0000 0x18413    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 15:47:27.810209+0000 0x1e785    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 16:32:00.817978+0000 0x22185    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-10 08:40:45.585492+0000 0x15a      Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): AA01102412155309367 8 0x781 0x5580 0x10, 3
2018-02-10 08:40:45.709848+0000 0x7a       Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
```


The USB Mass Storage Device Class (USBMSC) Identifier, usually the serial number of the device, can be found by doing a search for "USBMSC" in the systems using 10.8–10.11 in the `system.log`. In 10.12+ systems, it can be found in the Unified logs.

While the serial number may, in fact, be the one found on the suspect USB drive, it may not necessarily be unique, as the USBMSC message reminded us. An example of one such device is one that uses the letters A–F and numbers 0–9 as its identifier. The additional numbers are the vendor ID, Product ID, and version.

```
Jun  3 11:11:53 bit kernel[0]: USBMSC Identifier (non-unique):
FBF1011220504638 0x90c 0x1000 0x1100
```

The example USBMSC record shown above contains a USB device that is identified by the serial number `FBF1011220504638`. The additional data (using the System Information application) shows vendor/product data.

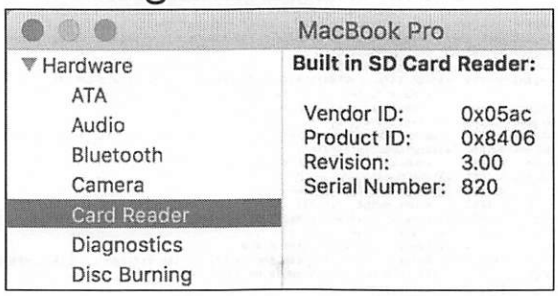
- Vendor ID: 0x090c (Silicon Motion, Inc).
- Product ID: 0x1000
- Version: 11.00

The website `http://usb-ids.gowdy.us/read/UD/` lists vendors by ID and products by ID. It may also link to a forum discussing the physical properties of the device.

```
Sarahs-MBP:FOR518 oompa$ log show galaga.logarchive/ --info --predicate 'eventMessage contains "USBMSC"'
==========
/Users/oompa/FOR518/galaga.logarchive
==========
log: warning: The log archive contains partial or missing metadata
Filtering the log data using "eventMessage CONTAINS "USBMSC""
Skipping debug messages, pass --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2018-02-07 03:49:50.586839-0500 0xbe       Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): AA011024121553093678 0x781 0x5580 0x10, 3
2018-02-07 03:49:50.816848-0500 0xbe       Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 04:39:37.971787-0500 0x46c3     Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 05:23:44.240661-0500 0x7e2a     Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 06:14:59.238835-0500 0xab1d     Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 06:57:30.434931-0500 0xe060     Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 08:05:12.588033-0500 0x15ae9    Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 09:20:47.558759-0500 0x16705    Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 14:37:00.136479+0000 0x18413    Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 15:47:27.810209+0000 0x1e785    Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 16:32:00.017978+0000 0x22185    Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
2018-02-10 08:40:45.585492+0000 0x15a      Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): AA011024121563093678 0x781 0x5580 0x10, 3
2018-02-10 08:40:45.709848+0000 0x7a       Default     0x0                  0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3
```

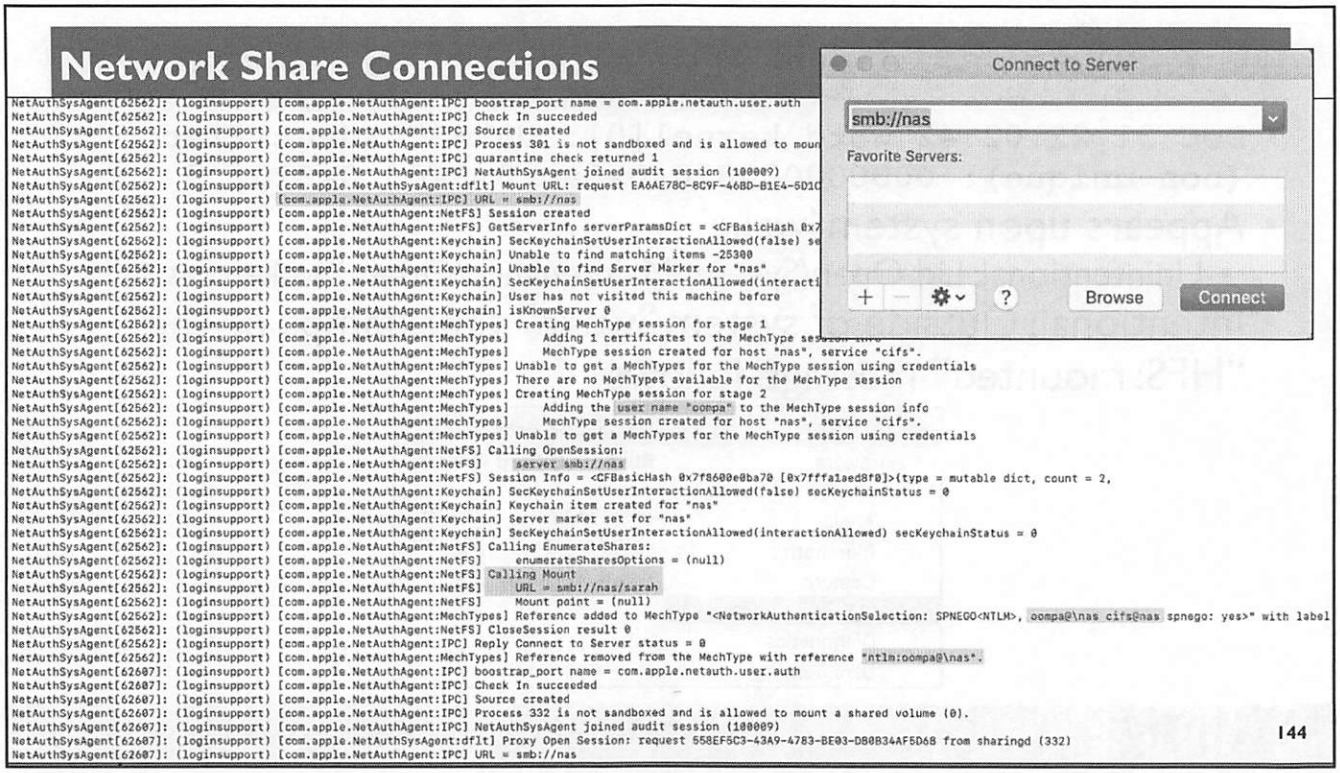
USBMSC Caveat: SD Card Reader

- `Dec 31 22:02:42 word kernel[0]: USBMSC Identifier (non-unique): 000000000820 0x5ac 0x8406 0x820, 3`
- Appears upon system 'wake'
  - Unintentional: Lid Open/System Maintenance/Other "wake reason"
- Intentional: Outside of system "wake" timestamps or the "HFS: mounted" message follows



If you are reviewing a Mac that has a built-in SD Card reader, you will notice that you get a large amount of "USBMSC" entries that appear like the example above. This is a powerful and misleading example of intentional and non-intentional log entries.

Some log entries are user initiated, such as plugging in a thumb drive to the system; while others, like this one, are just the operating system doing its normal logging. Experience and testing will show the analyst how to differentiate between these entries.



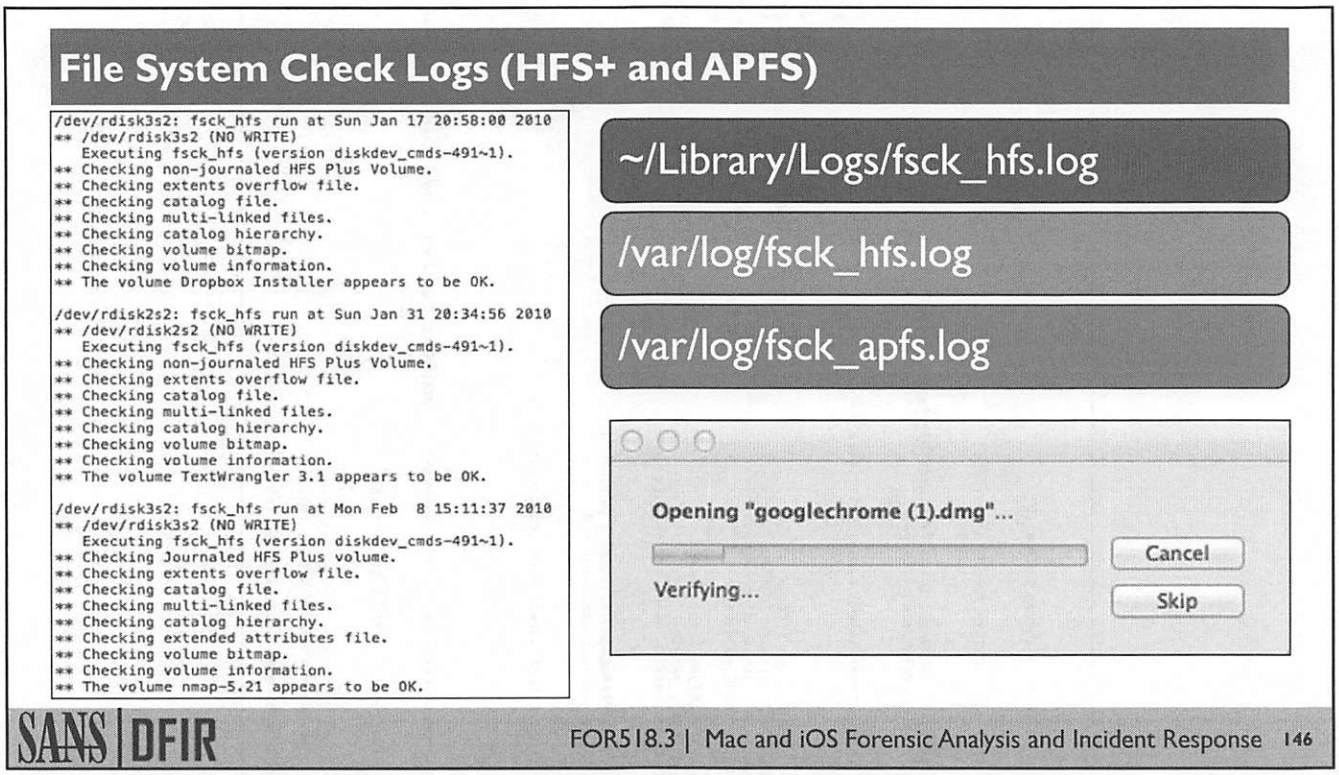
Unified logs can show connections to network shares. The example above was filtered using the keyword "NetAuthSysAgent" and "(loginsupport)".

It shows the network server (smb://nas), share (/sarah), and user (oompa) information.

This example was created using the "Connect to Server" functionality of Finder.

Older logs may use "afpfs" or "smbfs" to identify network shares.

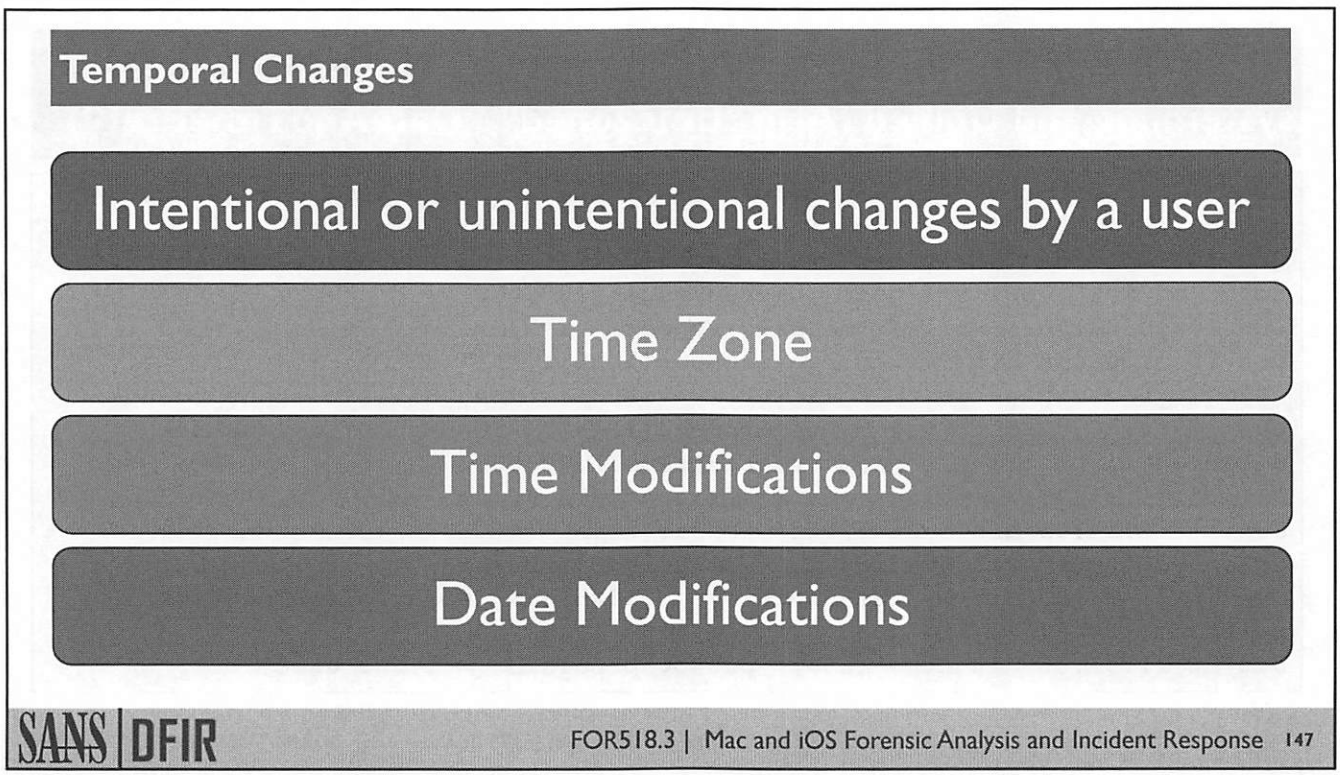
```
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:IPC] boostrap_port name = com.apple.netauth.user.auth
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:IPC] Check In succeeded
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:IPC] Source created
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:IPC] Process 301 is not sandboxed and is allowed to mount a shared volume (0).
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:IPC] quarantine check returned 1
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:IPC] NetAuthSysAgent joined audit session (100009)
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthSysAgent:dflt] Mount URL: request EA6AE78C-8C9F-46BD-B1E4-5D1C4AFB2F83 from Finder (301)
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:IPC] URL = smb://nas
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Session created
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS] GetServerInfo serverParamsDict = <CFBasicHash 0x7f8600c0b260 [0x7fffa1aed8f0]>{type = mutable dict, count = 6,
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] SecKeychainSetUserInteractionAllowed(false) secKeychainStatus = 0
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] Unable to find matching items -25300
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] Unable to find Server Marker for "nas"
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] SecKeychainSetUserInteractionAllowed(interactionAllowed) secKeychainStatus = 0
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] User has not visited this machine before
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] isKnownServer 0
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes] Creating MechType session for stage 1
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes]     Adding 1 certificates to the MechType session info
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes]     MechType session created for host "nas", service "cifs".
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes] Unable to get a MechTypes for the MechType session using credentials
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes] There are no MechTypes available for the MechType session
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes] Creating MechType session for stage 2
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes]     Adding the user name "oompa" to the MechType session info
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes]     MechType session created for host "nas", service "cifs".
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes] Unable to get a MechTypes for the MechType session using credentials
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Calling OpenSession:
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS]     server smb://nas
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Session Info = <CFBasicHash 0x7f8600e0ba70 [0x7fffa1aed8f0]>{type = mutable dict, count = 2,
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] SecKeychainSetUserInteractionAllowed(false) secKeychainStatus = 0
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] Keychain item created for "nas"
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] Server marker set for "nas"
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:Keychain] SecKeychainSetUserInteractionAllowed(interactionAllowed) secKeychainStatus = 0
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Calling EnumerateShares:
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS]     enumerateSharesOptions = (null)
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Calling Mount
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS]     URL = smb://nas/sarah
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS]     Mount point = (null)
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes] Reference added to MechType "<NetworkAuthenticationSelection: SPNEGO<NTLM>, oompa@\nas cifs@nas spnego: yes>" with label
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:NetFS] CloseSession result 0
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:IPC] Reply Connect to Server status = 0
NetAuthSysAgent[62562]: (loginsupport) [com.apple.NetAuthAgent:MechTypes] Reference removed from the MechType with reference "ntlm:oompa@\nas".
NetAuthSysAgent[62607]: (loginsupport) [com.apple.NetAuthAgent:IPC] boostrap_port name = com.apple.netauth.user.auth
NetAuthSysAgent[62607]: (loginsupport) [com.apple.NetAuthAgent:IPC] Check In succeeded
NetAuthSysAgent[62607]: (loginsupport) [com.apple.NetAuthAgent:IPC] Source created
NetAuthSysAgent[62607]: (loginsupport) [com.apple.NetAuthAgent:IPC] Process 332 is not sandboxed and is allowed to mount a shared volume (0).
NetAuthSysAgent[62607]: (loginsupport) [com.apple.NetAuthAgent:IPC] NetAuthSysAgent joined audit session (100009)
NetAuthSysAgent[62607]: (loginsupport) [com.apple.NetAuthSysAgent:dflt] Proxy Open Session: request 558EF5C3-43A9-4783-BE03-D80B34AF5D68 from sharingd (332)
NetAuthSysAgent[62607]: (loginsupport) [com.apple.NetAuthAgent:IPC] URL = smb://nas
```



Each user may have their own `fsck_hfs.log`.

This log keeps track of mounted volumes that execute the "verifying" process when a user opens a disk image file. This can show programs that the user may have installed, when the disk image was opened, and the location on the `/dev/` tree where it may be found (i.e., `/dev/rdisk3s2`).

There may also be system-wide fsck_hfs.log and fsck_apfs.log files as well.



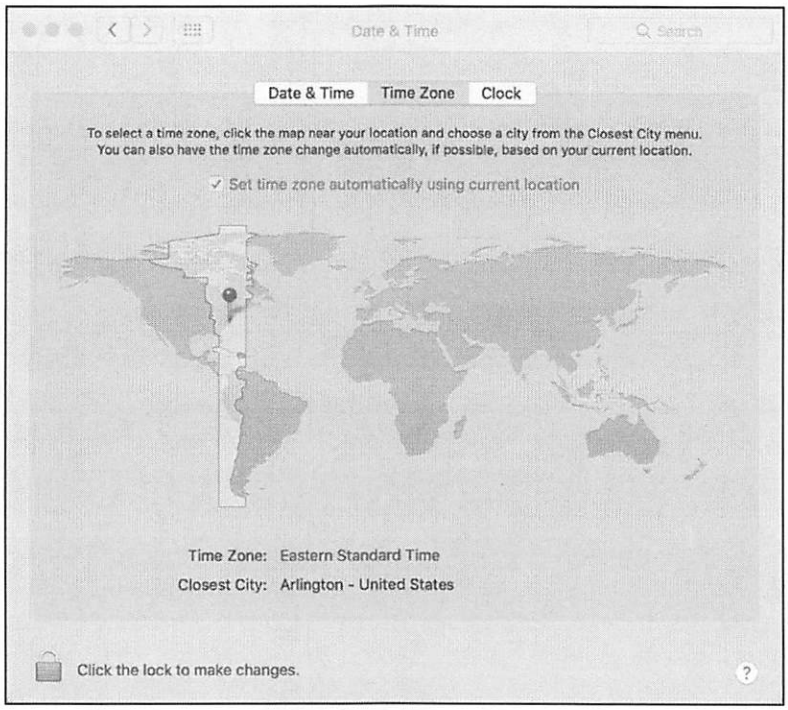
A user may make intentional changes to the system time or date. They might, for example, be traveling and need to change time zones. A user may also make changes to the system time or date to throw a temporal investigation off.

Time Changes: Going Back in Time
system.log and Unified (Also "timezoned", "DateAndTimePref")

```
Jun 16 14:50:56 bit System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] Invalidating _selectedCityLayer
Jun 16 14:50:56 bit System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] all good  cachedValue:1.000000
Jun 16 14:50:56: --- last message repeated 1 time ---
Jun 16 14:50:56 bit System Preferences[1828]: **** ERROR: -
[GEOCityPickerView _bindPublicToPrivateProperties] UI is already bounded
Jun 16 14:50:59 bit System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] all good  cachedValue:1.000000
Jun 16 11:51:05: --- last message repeated 4 times ---
Jun 16 11:51:05 bit System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] Invalidating _selectedCityLayer
Jun 16 11:51:05 bit System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] all good  cachedValue:1.000000
Jun 16 11:51:06 bit ntpd[1848]: proto: precision = 1.000 usec
```


A user might attempt to hide or obfuscate activity by changing the system time.

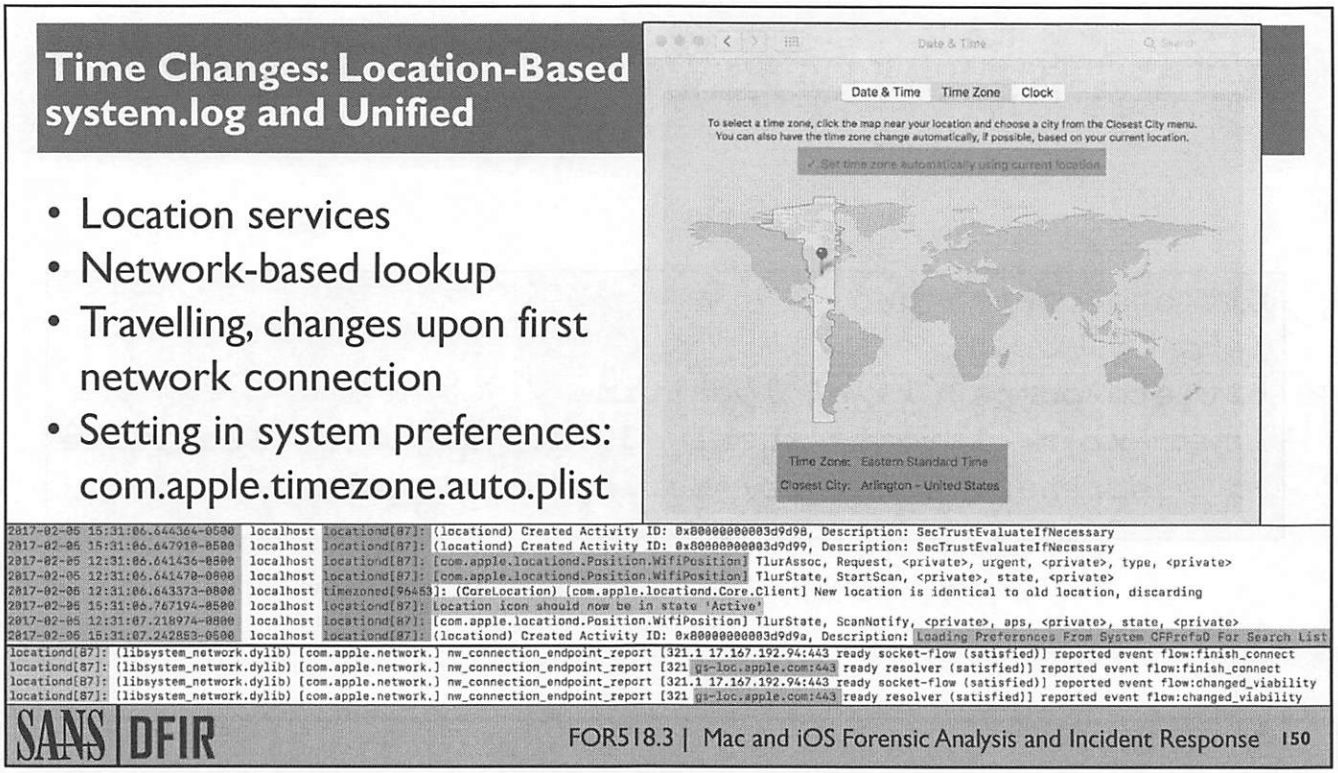
In the screenshot above, the system time was changed to go back in time three hours. The highlighted timestamps in the `system.log` file show that the time was changed using the System Preferences (Date & Time) preference panel. The `GEOWorldTimeZoneView` keyword should indicate that the user is interfacing with the "Time Zone" preferences panel.



Time Changes: Time Zone—/etc/localtime

```
bit:etc oompa$ pwd
/etc
bit:etc oompa$ ls -l localtime
lrwxr-xr-x  1 root  wheel  39 Jun 16 11:51 localtime
-> /usr/share/zoneinfo/America/Los_Angeles
```


The `localtime` symlink is updated when the time zone is changed to reflect the new time zone. In the example shown, the time zone was last changed to Los Angeles time on June 16.



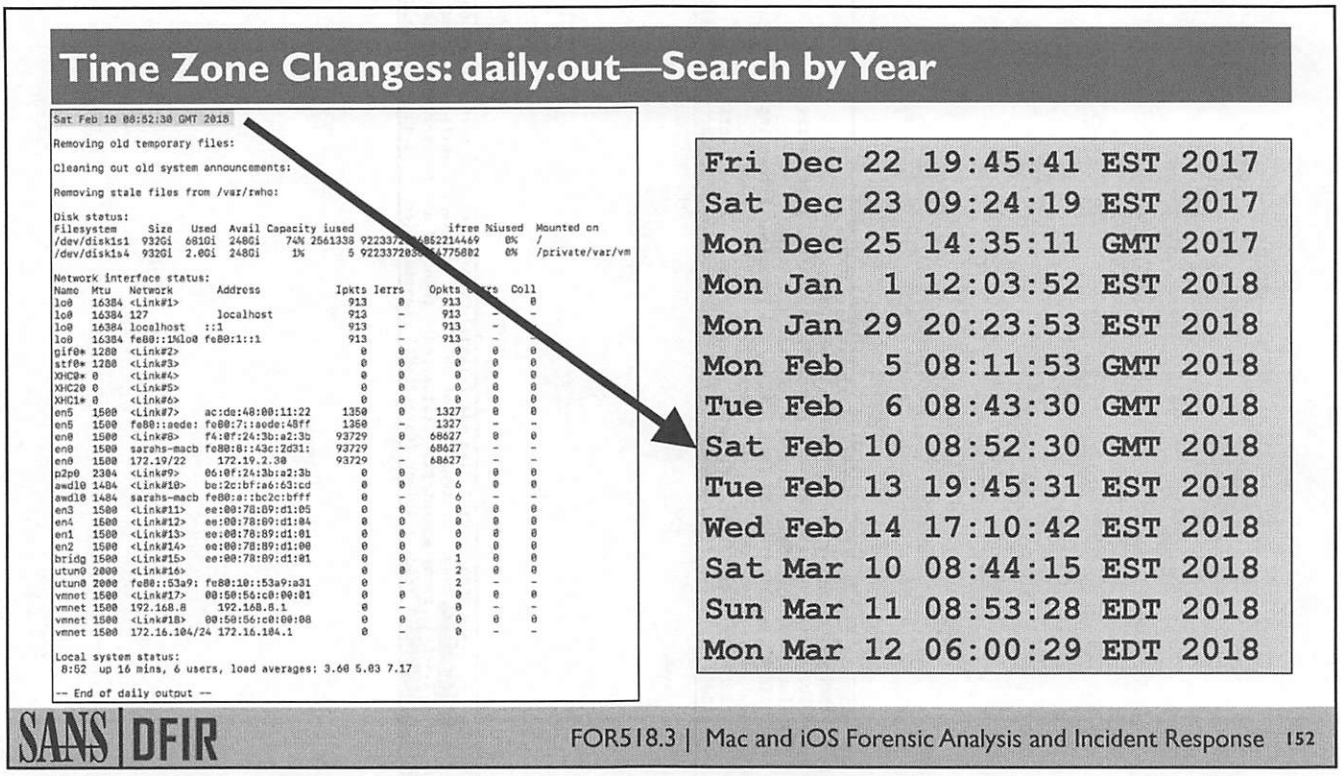
If location-based services are used, the macOS system will perform a lookup of where the device is (assuming it is connected to the network) in an attempt to determine its location.

In the logs, search for "`location`" (the location services daemon) or "`timezoned`" (the time zone daemon). Other indicators in the logs may suggest the time is being changed, such as the logging timestamp and call outs to `gs-loc.apple.com`, Apple's Location server.

You can find the binary setting for this feature in the `com.apple.timezone.auto.plist` in `/Library/Preferences/`.

```
2017-02-05 15:31:06.644364-0500  localhost locationd[87]: (locationd) Created Activity ID: 0x80000000003d9d98, Description: SecTrustEvaluateIfNecessary
2017-02-05 15:31:06.647910-0500  localhost locationd[87]: (locationd) Created Activity ID: 0x80000000003d9d99, Description: SecTrustEvaluateIfNecessary
2017-02-05 12:31:06.641436-0800  localhost locationd[87]: [com.apple.locationd.Position.WifiPosition] TlurAssoc, Request, <private>, urgent, <private>, type, <private>
2017-02-05 12:31:06.641470-0800  localhost locationd[87]: [com.apple.locationd.Position.WifiPosition] TlurState, StartScan, <private>, state, <private>
2017-02-05 12:31:06.643373-0800  localhost timezoned[96453]: (CoreLocation) [com.apple.locationd.Core.Client] New location is identical to old location, discarding
2017-02-05 15:31:06.767194-0500  localhost locationd[87]: Location icon should now be in state 'Active'
2017-02-05 12:31:07.218974-0800  localhost locationd[87]: [com.apple.locationd.Position.WifiPosition] TlurState, ScanNotify, <private>, aps, <private>, state, <private>
2017-02-05 15:31:07.242853-0500  localhost locationd[87]: (locationd) Created Activity ID: 0x80000000003d9d9a, Description: Loading Preferences From System CFPrefsD For Search List
```

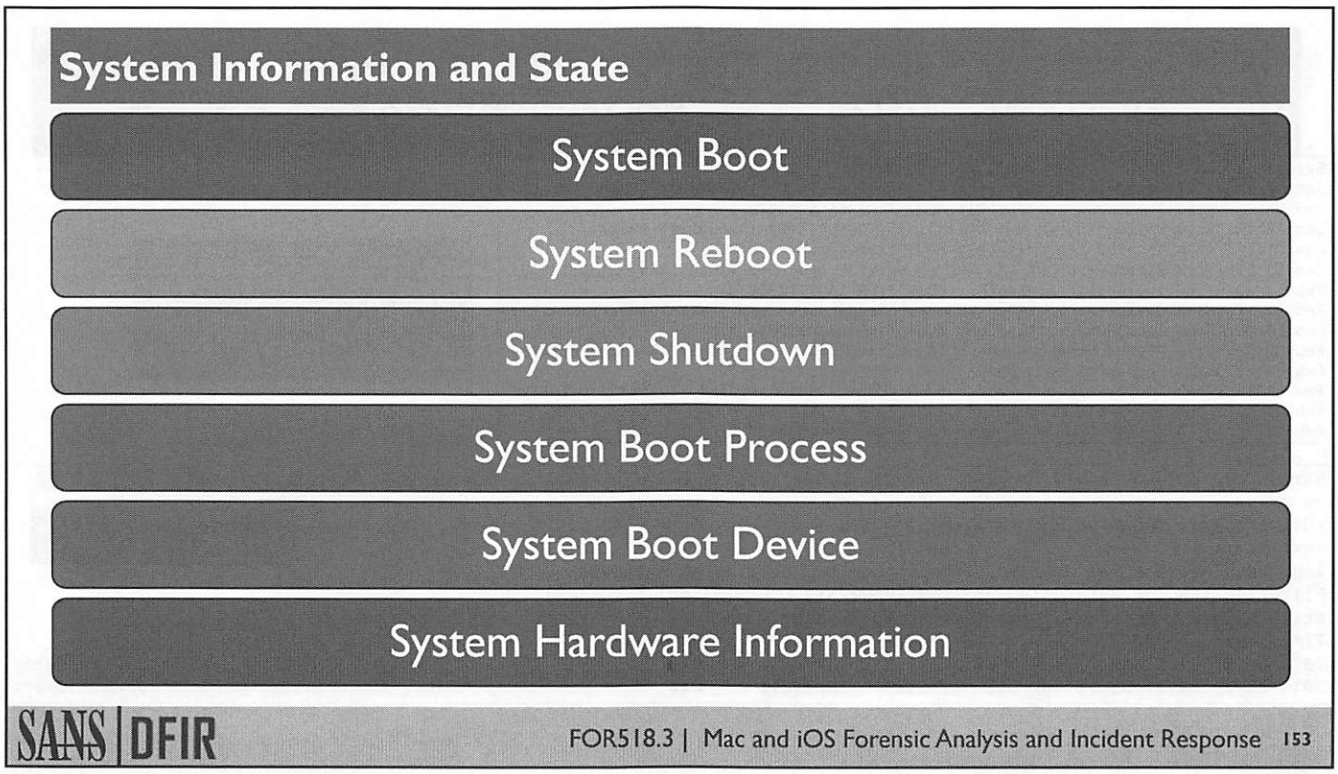
```
locationd[87]: (libsystem_network.dylib) [com.apple.network.] nw_connection_endpoint_report [321.1 17.167.192.94:443 ready socket-flow (satisfied)] reported event flow:finish_connect
locationd[87]: (libsystem_network.dylib) [com.apple.network.] nw_connection_endpoint_report [321 gs-loc.apple.com:443 ready resolver (satisfied)] reported event flow:finish_connect
locationd[87]: (libsystem_network.dylib) [com.apple.network.] nw_connection_endpoint_report [321.1 17.167.192.94:443 ready socket-flow (satisfied)] reported event flow:changed_viability
locationd[87]: (libsystem_network.dylib) [com.apple.network.] nw_connection_endpoint_report [321 gs-loc.apple.com:443 ready resolver (satisfied)] reported event flow:changed_viability
```



Time zone changes can also be shown in the `daily.out` file.

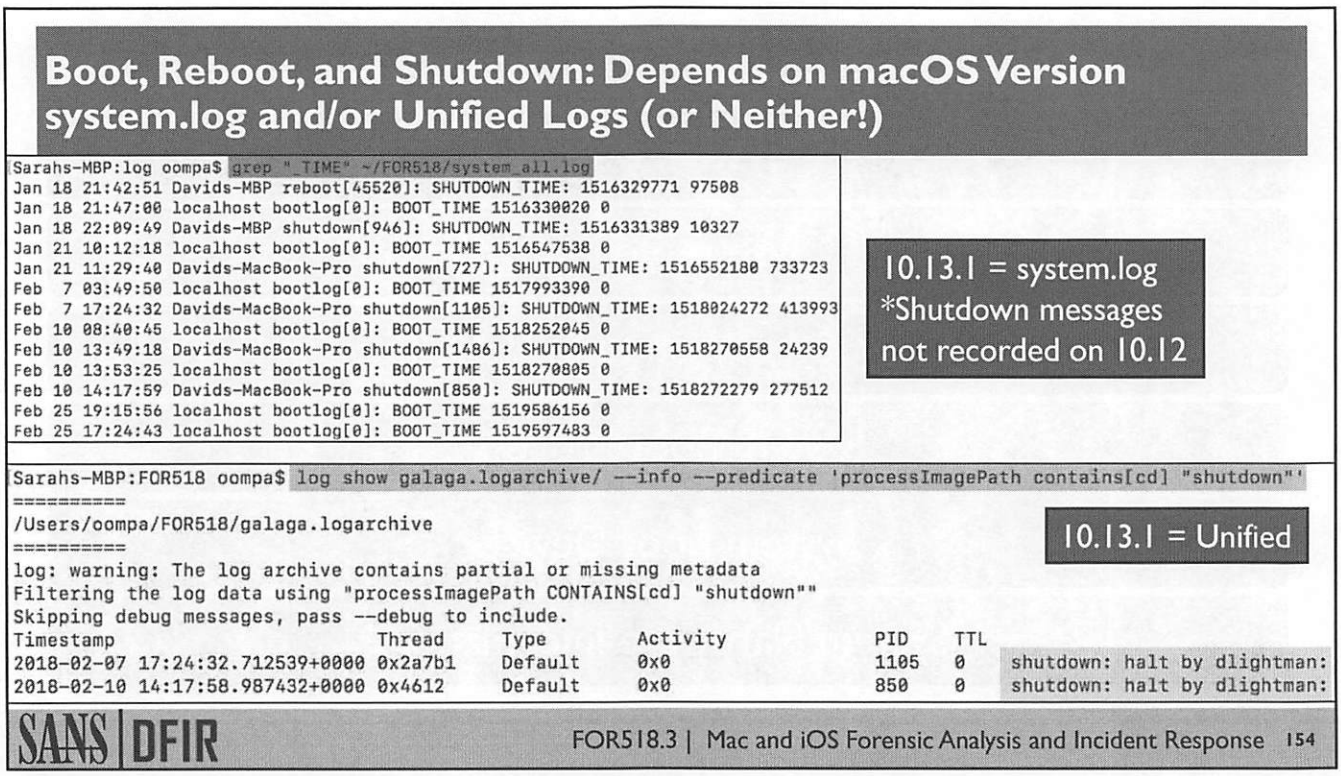
In the example above, a search was completed for the partial year with a space "`201`" so it would show both 2017 and 2018.

Note the changes from EST -> GMT -> EST -> GMT -> EST -> EDT



The logs can tell us a lot about how and when a system was used. In a temporal context, we can see when a system was powered on, when it was in a sleep state, and when it was shut down.

The hardware configuration of the system can be determined to see if the system was upgraded or if the boot drive changed systems.



Knowing when the system boots or when it is shut down can help with the timeline aspect of some investigations. On 10.13.1, the `system.log` contains entries for "`BOOT_TIME`" and "`SHUTDOWN_TIME`". This includes a Unix epoch timestamp as well.

The unified logs record which user account shut down ("`halt`") the system. If a reboot occurs, it will have a slightly different message—"`shutdown: reboot by <user>`".

*Note: 10.12.0–10.12.2—The Shutdown messages do not appear to get recorded in either log.

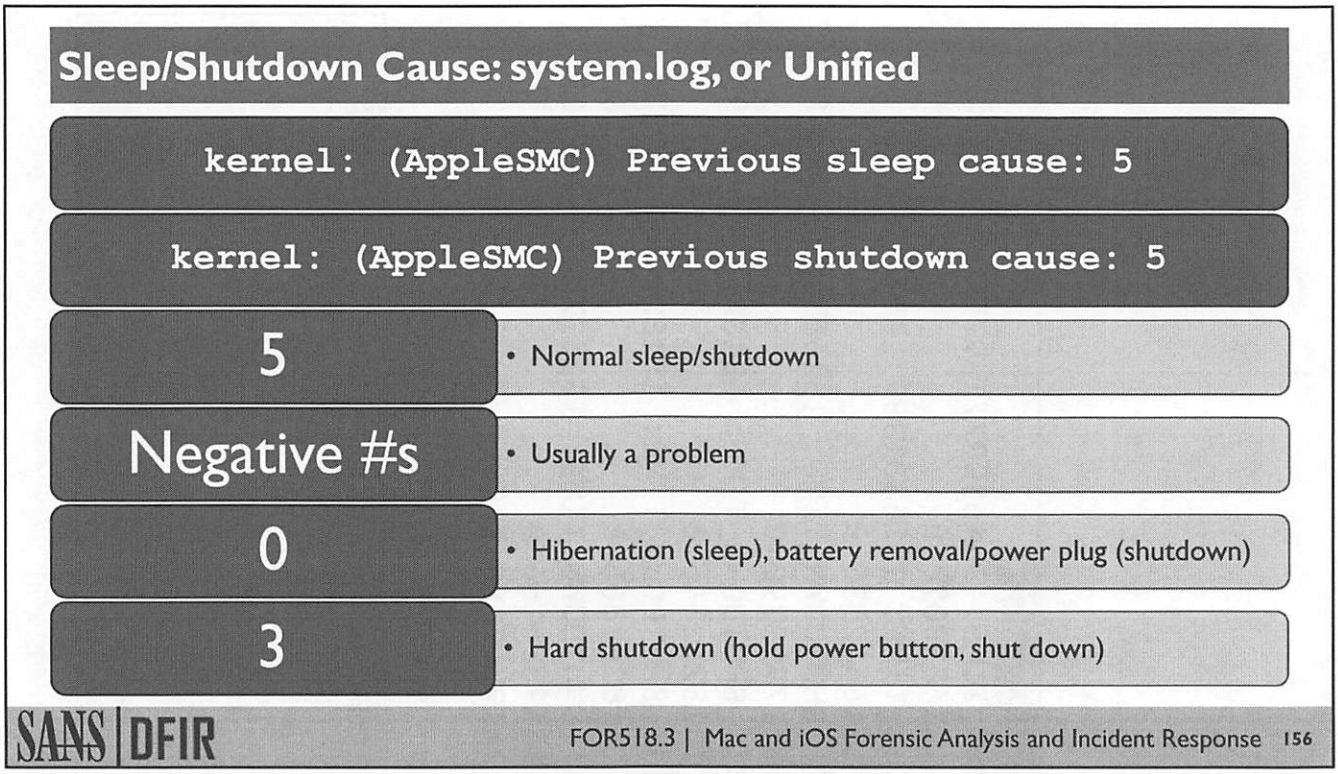
Older examples are below from the system.log.

```
May 27 20:02:14 bit shutdown[42801]: halt by oompa:
May 27 20:02:14 bit shutdown[42801]: SHUTDOWN_TIME: 1338163334 903688
May 28 15:20:06 bit shutdown[2421]: halt by oompa:
May 28 15:20:06 bit shutdown[2421]: SHUTDOWN_TIME: 1338232806 702175
Jun  9 09:25:33 bit shutdown[25868]: halt by oompa:
Jun  9 09:25:33 bit shutdown[25868]: SHUTDOWN_TIME: 1339248333 887656
Jun  9 10:15:24 bit shutdown[546]: reboot by oompa:
Jun  9 10:15:24 bit shutdown[546]: SHUTDOWN_TIME: 1339251324 30856
Jun  9 10:21:53 bit shutdown[309]: halt by oompa:
Jun  9 10:21:53 bit shutdown[309]: SHUTDOWN_TIME: 1339251713 535787
```

```
May  9 16:28:48 localhost bootlog[0]: BOOT_TIME 1336606128 0
May 10 16:40:27 localhost bootlog[0]: BOOT_TIME 1336682427 0
May 12 11:32:16 localhost bootlog[0]: BOOT_TIME 1336836736 0
May 27 20:02:41 localhost bootlog[0]: BOOT_TIME 1338163361 0
May 28 15:22:30 localhost bootlog[0]: BOOT_TIME 1338232950 0
Jun  9 09:27:05 localhost bootlog[0]: BOOT_TIME 1339248425 0
Jun  9 10:15:56 localhost bootlog[0]: BOOT_TIME 1339251356 0
Jun  9 10:33:39 localhost bootlog[0]: BOOT_TIME 1339252419 0
Jun  9 09:27:05 localhost bootlog[0]: BOOT_TIME 1339248425 0
Jun  9 10:15:56 localhost bootlog[0]: BOOT_TIME 1339251356 0
Jun  9 10:33:39 localhost bootlog[0]: BOOT_TIME 1339252419 0
Jun 10 13:33:56 localhost bootlog[0]: BOOT_TIME 1339349636 0
Jun 12 10:16:35 localhost bootlog[0]: BOOT_TIME 1339510595 0
```

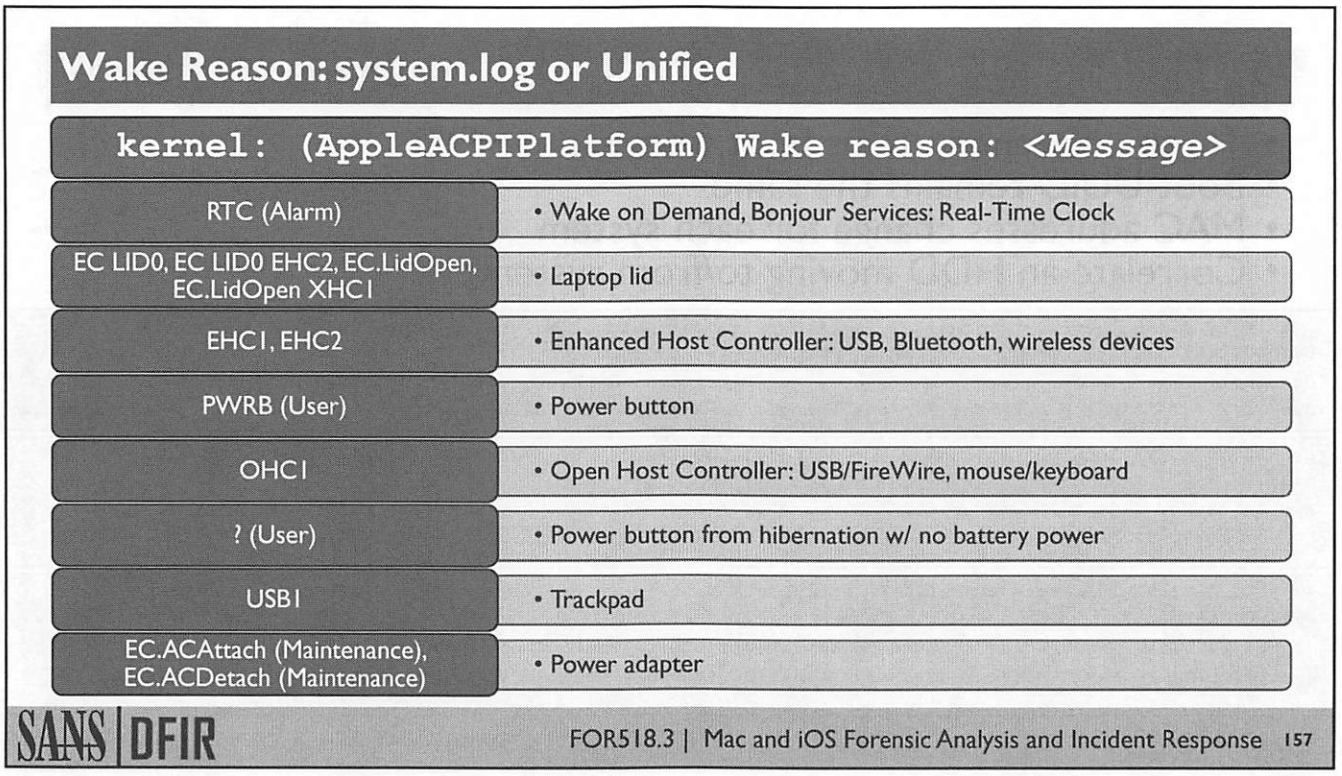
```
Sarahs-MBP:log oompa$ grep "_TIME" ~/FOR518/system_all.log
Jan 18 21:42:51 Davids-MBP reboot[45520]: SHUTDOWN_TIME: 1516329771 97508
Jan 18 21:47:00 localhost bootlog[0]: BOOT_TIME 1516330020 0
Jan 18 22:09:49 Davids-MBP shutdown[946]: SHUTDOWN_TIME: 1516331389 10327
Jan 21 10:12:18 localhost bootlog[0]: BOOT_TIME 1516547538 0
Jan 21 11:29:40 Davids-MacBook-Pro shutdown[727]: SHUTDOWN_TIME: 1516552180 733723
Feb  7 03:49:50 localhost bootlog[0]: BOOT_TIME 1517993390 0
Feb  7 17:24:32 Davids-MacBook-Pro shutdown[1105]: SHUTDOWN_TIME: 1518024272 413993
Feb 10 08:40:45 localhost bootlog[0]: BOOT_TIME 1518252045 0
Feb 10 13:49:18 Davids-MacBook-Pro shutdown[1486]: SHUTDOWN_TIME: 1518270558 24239
Feb 10 13:53:25 localhost bootlog[0]: BOOT_TIME 1518270805 0
Feb 10 14:17:59 Davids-MacBook-Pro shutdown[850]: SHUTDOWN_TIME: 1518272279 277512
Feb 25 19:15:56 localhost bootlog[0]: BOOT_TIME 1519586156 0
Feb 25 17:24:43 localhost bootlog[0]: BOOT_TIME 1519597483 0
```

```
Sarahs-MBP:FOR518 oompa$ log show galaga.logarchive/ --info --predicate 'processImagePath contains[cd] "shutdown"'
==========
/Users/oompa/FOR518/galaga.logarchive
==========
log: warning: The log archive contains partial or missing metadata
Filtering the log data using "processImagePath CONTAINS[cd] "shutdown""
Skipping debug messages, pass --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2018-02-07 17:24:32.712539+0000 0x2a7b1    Default     0x0                  1105   0    shutdown: halt by dlightman:
2018-02-10 14:17:58.987432+0000 0x4612     Default     0x0                  850    0    shutdown: halt by dlightman:
```



The system records reasons the computer has been put to sleep, which is known as "`Sleep Cause`" or "`Shutdown Cause`".

The cause that should be most prevalent on a working system should be cause number "`5`".



The system records the reasons the computer has awoken, known as "`Wake Reason`". Outlined above are some of the reasons.

System Boot: MAC Addresses—system.log [10.11-]

- Three different systems boot from same HDD
- Boot-UUID remains the same
- MAC addresses change for each system
- Correlate an HDD moving to/from systems

```
Jun 14 17:41:59 localhost kernel[0]: rooting via boot-uuid from /chosen: 1FDCF218-B7EB-3BAC-9AD6-
8498D0E2EA9D
Jun 14 17:42:22 Sarah-Edwardss-MacBook kernel[0]: yukon: Ethernet address 00:19:e3:3c:cb:7e
Jun 14 17:42:22 Sarah-Edwardss-MacBook kernel[0]: AirPort_AthrFusion21: Ethernet address 00:1b:63:c3:8d:1a
```

```
Jun 14 19:50:26 localhost kernel[0]: rooting via boot-uuid from /chosen: 1FDCF218-B7EB-3BAC-9AD6-
8498D0E2EA9D
Jun 14 19:50:53 Sarah-Edwardss-MacBook kernel[0]: AirPort_Brcm4331: Ethernet address 28:cf:da:04:84:77
Jun 14 19:50:53 Sarah-Edwardss-MacBook kernel[0]: BCM5701Enet: Ethernet address 3c:07:54:03:65:20
```

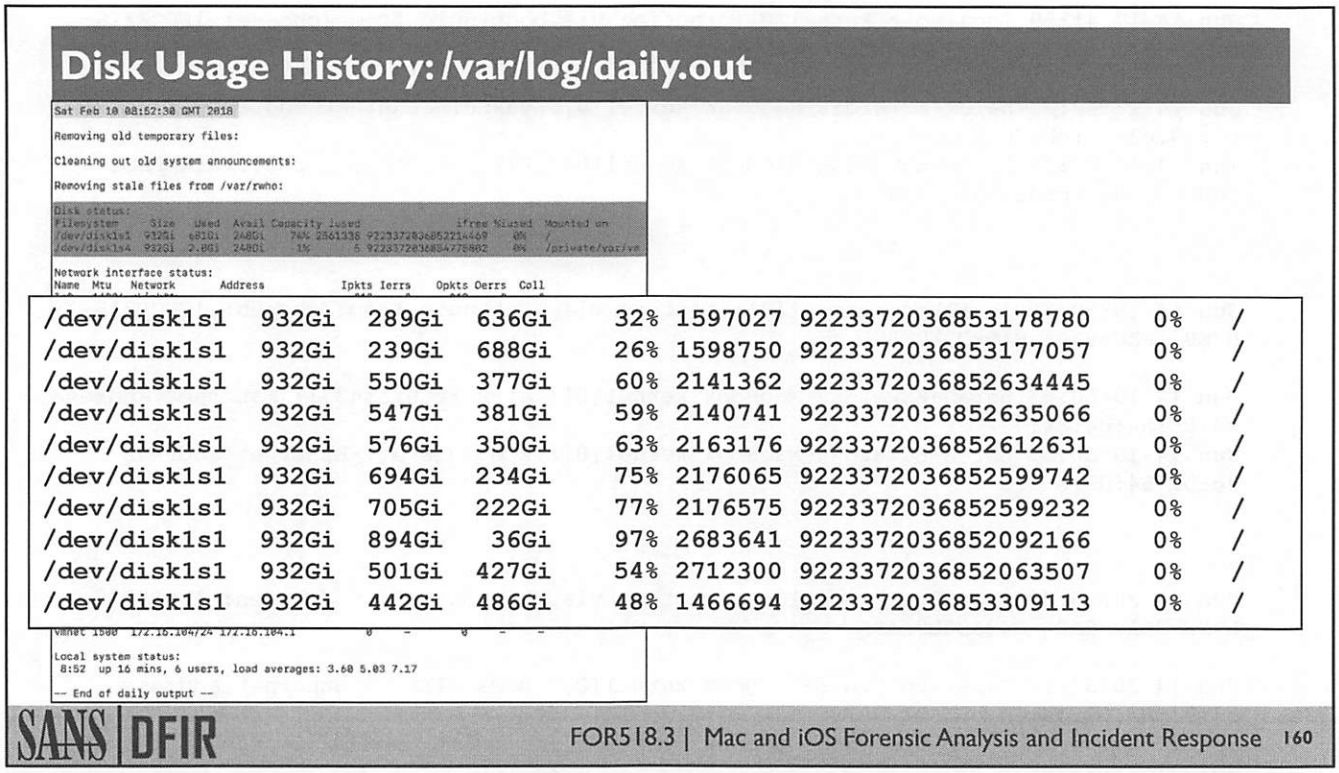
```
Jun 14 20:33:38 localhost kernel[0]: rooting via boot-uuid from /chosen: 1FDCF218-B7EB-3BAC-9AD6-
8498D0E2EA9D
Jun 14 20:34:12 Sarah-Edwardss-MacBook kernel[0]: BCM5701Enet: Ethernet address c4:2c:03:09:ca:fd
Jun 14 20:34:12 Sarah-Edwardss-MacBook kernel[0]: AirPort_Brcm43224: Ethernet address 90:27:e4:f8:e6:5f
```


One way to correlate if a hard drive has been moved from system to system is to check for the MAC addresses.

In the example, each system has two network cards (one wireless, one wired). Immediately, we can probably rule out the MacBook Air as one of the systems, as these only have one wireless NIC by default. The message includes a hint as to what types of network cards are installed on the systems.

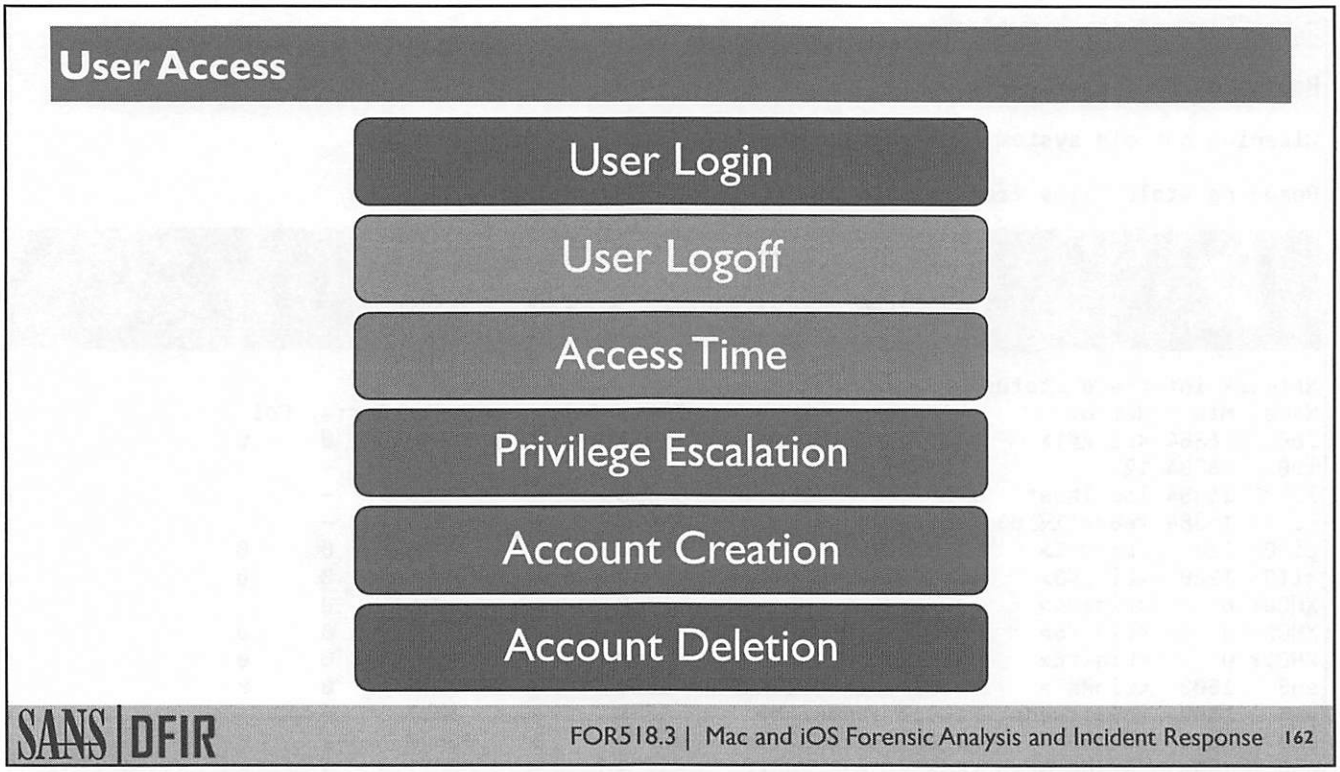
- System 1
  - `Yukon`: Yukon network card
  - `AirPort_AthrFusion21`: Atheros network card
- System 2
  - `AirPort_Brcm4331`: Broadcom network card
  - `BCM5701Enet`: Broadcom network card
- System 3
  - `BCM5701Enet`: Broadcom network card
  - `Airport_Brcm43224`: Broadcom network card

```
Jun 14 17:41:59 localhost kernel[0]: rooting via boot-uuid from /chosen: 1FDCF218-
B7EB-3BAC-9AD6-8498D0E2EA9D

Jun 14 17:42:22 Sarah-Edwardss-MacBook kernel[0]: yukon: Ethernet address
00:19:e3:3c:cb:7e
Jun 14 17:42:22 Sarah-Edwardss-MacBook kernel[0]: AirPort_AthrFusion21: Ethernet
address 00:1b:63:c3:8d:1a



Jun 14 19:50:26 localhost kernel[0]: rooting via boot-uuid from /chosen: 1FDCF218-
B7EB-3BAC-9AD6-8498D0E2EA9D

Jun 14 19:50:53 Sarah-Edwardss-MacBook kernel[0]: AirPort_Brcm4331: Ethernet address
28:cf:da:04:84:77
Jun 14 19:50:53 Sarah-Edwardss-MacBook kernel[0]: BCM5701Enet: Ethernet address
3c:07:54:03:65:20



Jun 14 20:33:38 localhost kernel[0]: rooting via boot-uuid from /chosen: 1FDCF218-
B7EB-3BAC-9AD6-8498D0E2EA9D

Jun 14 20:34:12 Sarah-Edwardss-MacBook kernel[0]: BCM5701Enet: Ethernet address
c4:2c:03:09:ca:fd
Jun 14 20:34:12 Sarah-Edwardss-MacBook kernel[0]: AirPort_Brcm43224: Ethernet address
90:27:e4:f8:e6:5f
```

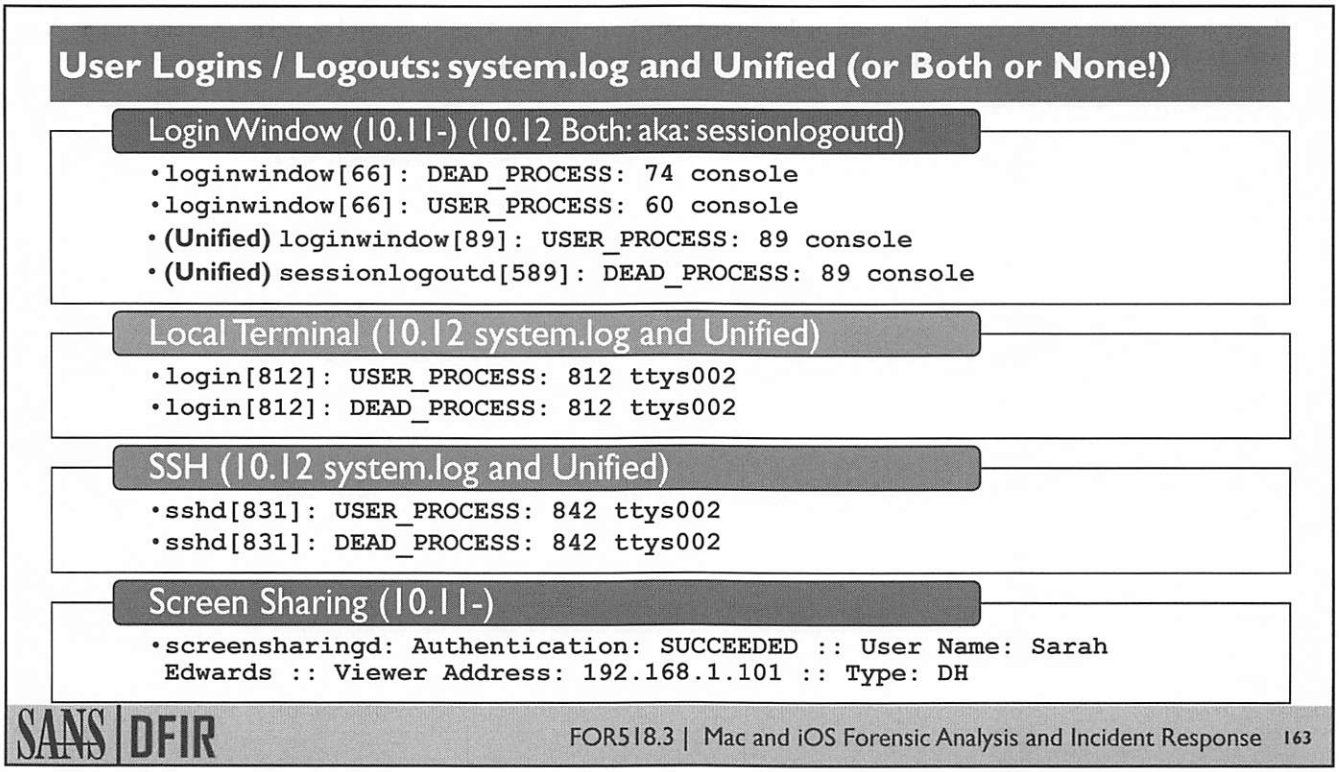


The disk usage for a system might be useful to see how much disk space a system uses over time. The `daily.out` is one of the log files associated with the Unix maintenance scripts (along with `weekly.out` and `monthly.out`), which records "`Disk Status`". We can "`grep`" this log file for the specific disk (`/dev/disk1s1`), as shown in the example above.

```
Sat Feb 10 08:52:30 GMT 2018

Removing old temporary files:

Cleaning out old system announcements:

Removing stale files from /var/rwho:

Disk status:
Filesystem     Size   Used  Avail Capacity iused               ifree %iused  Mounted on
/dev/disk1s1  932Gi  681Gi  248Gi    74% 2561338 9223372036852214469    0%   /
/dev/disk1s4  932Gi  2.0Gi  248Gi     1%       5 9223372036854775802    0%   /private/var/vm

Network interface status:
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16384 <Link#1>                         913     0      913     0     0
lo0   16384 127           localhost          913     -      913     -     -
lo0   16384 localhost   ::1                  913     -      913     -     -
lo0   16384 fe80::1%lo0 fe80:1::1            913     -      913     -     -
gif0* 1280  <Link#2>                           0     0        0     0     0
stf0* 1280  <Link#3>                           0     0        0     0     0
XHC0* 0     <Link#4>                           0     0        0     0     0
XHC20 0     <Link#5>                           0     0        0     0     0
XHC1* 0     <Link#6>                           0     0        0     0     0
en5   1500  <Link#7>    ac:de:48:00:11:22   1350     0     1327     0     0
en5   1500  fe80::aede: fe80:7::aede:48ff    1350     -     1327     -     -
en0   1500  <Link#8>    f4:0f:24:3b:a2:3b  93729     0    68627     0     0
en0   1500  sarahs-macb fe80:8::43c:2d31:   93729     -    68627     -     -
en0   1500  172.19/22     172.19.2.30       93729     -    68627     -     -
p2p0  2304  <Link#9>    06:0f:24:3b:a2:3b       0     0        0     0     0
awdl0 1484  <Link#10>   be:2c:bf:a6:63:cd       0     0        6     0     0
awdl0 1484  sarahs-macb fe80:a::bc2c:bfff       0     -        6     -     -
en3   1500  <Link#11>   ee:00:78:89:d1:05       0     0        0     0     0
en4   1500  <Link#12>   ee:00:78:89:d1:04       0     0        0     0     0
en1   1500  <Link#13>   ee:00:78:89:d1:01       0     0        0     0     0
en2   1500  <Link#14>   ee:00:78:89:d1:00       0     0        0     0     0
bridg 1500  <Link#15>   ee:00:78:89:d1:01       0     0        1     0     0
utun0 2000  <Link#16>                           0     0        2     0     0
utun0 2000  fe80::53a9: fe80:10::53a9:a31       0     -        2     -     -
vmnet 1500  <Link#17>   00:50:56:c0:00:01       0     0        0     0     0
vmnet 1500  192.168.8     192.168.8.1          0     -        0     -     -
vmnet 1500  <Link#18>   00:50:56:c0:00:08       0     0        0     0     0
vmnet 1500  172.16.104/24 172.16.104.1         0     -        0     -     -

Local system status:
 8:52  up 16 mins, 6 users, load averages: 3.60 5.03 7.17

-- End of daily output --
```



Logs can tell a lot about how each user makes use of a system—when they logged in or logged off, if they have privileged access, and how long they were on the system.



User logins and logouts can help in determining the usage of the system, who are the main actors on the system, and who might not be allowed onto the system but is using it anyway!

OS X systems may be logged into by a variety of means, four of which are specified here.

- Login Window: Via the logon GUI
- Local Terminal: Via the Terminal program
- SSH: OpenSSH
- Screen Sharing: A native VNC program

Performing a search for logins can be accomplished by using the term "`_PROCESS`", except for Screen Sharing, which can be found by searching for "`screensharingd`".

Each login process will be marked with "`USER_PROCESS`" and the process ID, while the logoff will be shown as "`DEAD_PROCESS`" and the matching process ID.

The Login Window example shows a login (PID 74) and logoff (PID 60) for two different sessions, labeled as "`console`". Note this type uses the "`loginwindow`" process. On 10.12, the login entries appear the same in the system.log, while the logout entries are in the Unified logs under the "`sessionlogoutd`" process.

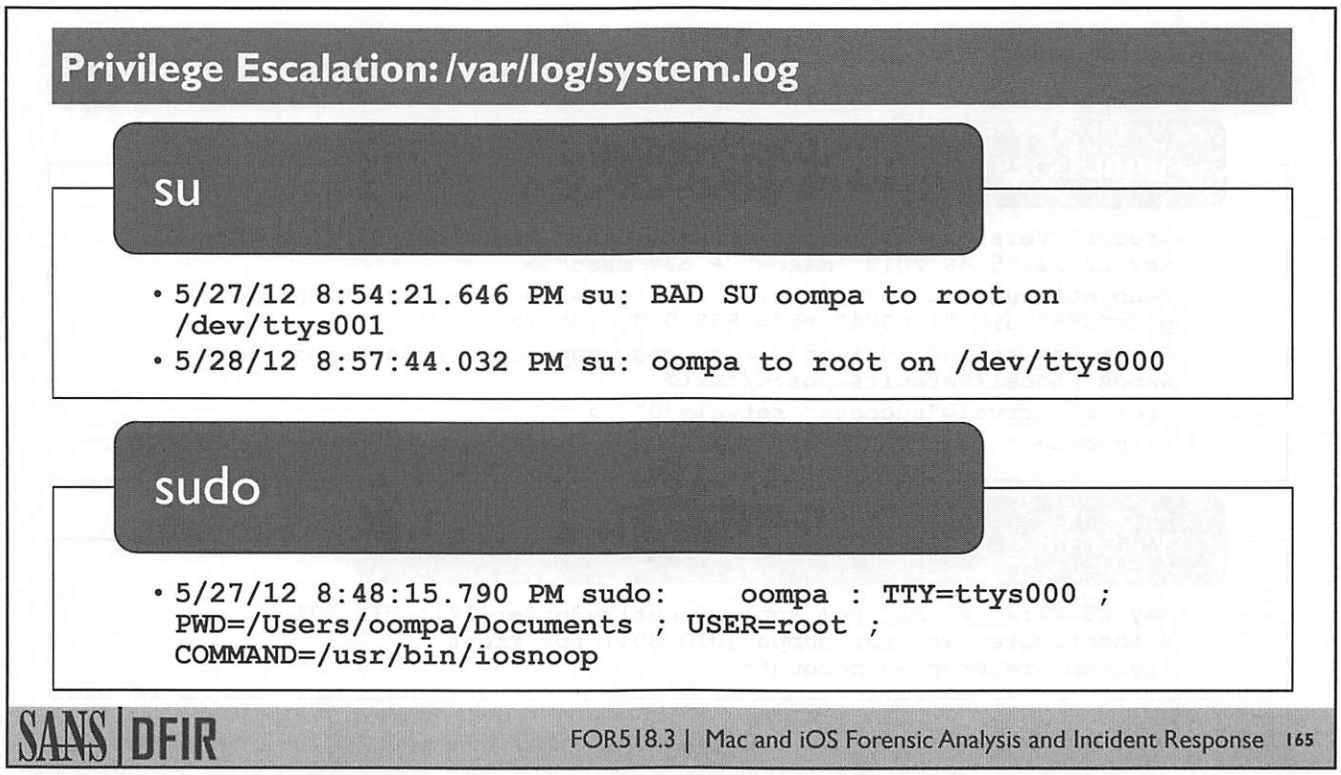
In the Local Terminal example, we can see the one login/logoff (PID 812). The terminal windows have been opened, and `ttys002` was used. Note this type uses the "`login`" process.

SSH uses the "`sshd`" process to record logins and logoffs. In the example, one login/logoff pair for PID 842 is shown using the `ttys002` terminal window.

On 10.12, the terminal login and sshd entries are separated. The USER_PROCESS is in the system.log, while the matching DEAD_PROCESS is in the Unified logs.

Screen Sharing logins show an entry like the one shown in the example above. It records the username (may be Full Name or username) and the IP address where the connection is coming from. These entries do not appear on 10.12.

These login/logout messages can be found in the "`system.log`" or the Apple System Logs. These events in the ASL logs contain more detailed information, such as which user logged in.

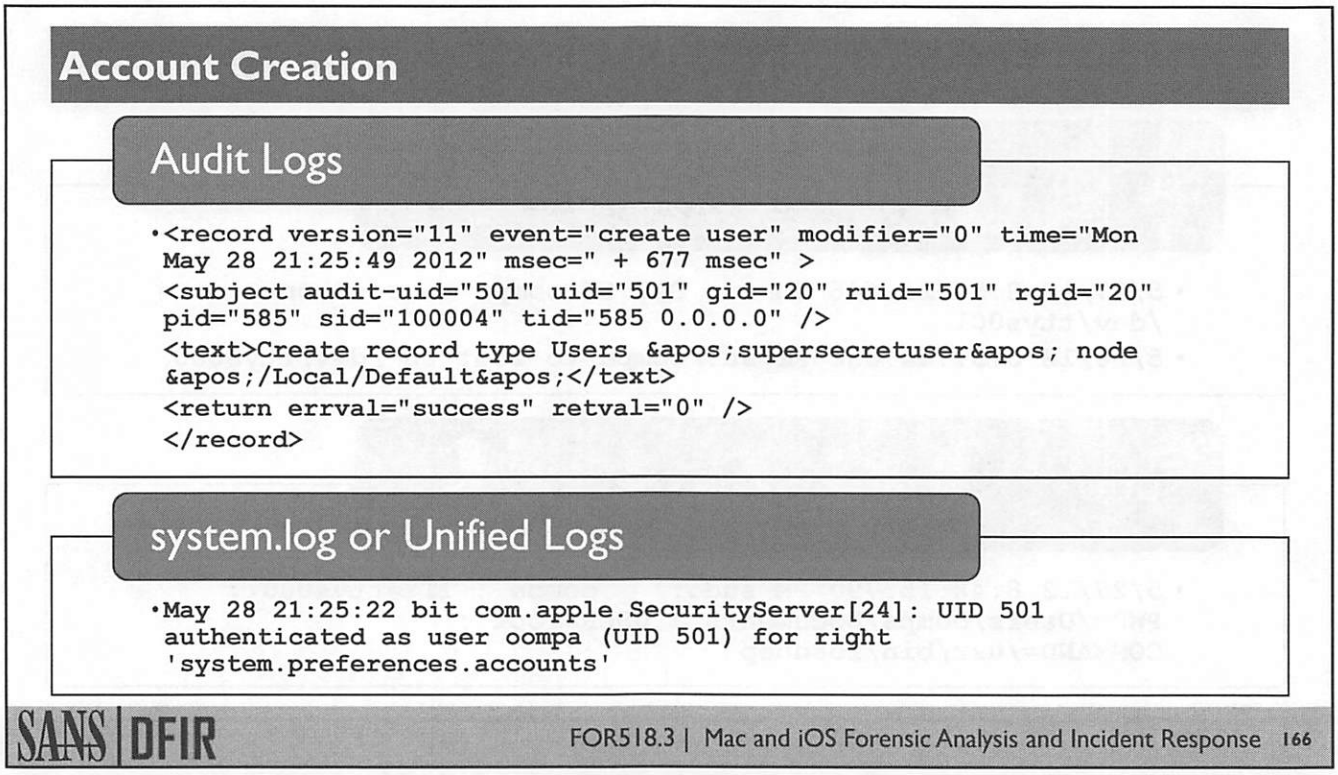


Privilege escalation is needed when a Standard user needs to perform root-level actions. The `su` command can be used to log in as root (or another user) for a session, while the `sudo` command will run a command as root for five minutes (this time can be changed in the `/etc/sudoers` configuration file).

The `su` example shows an entry that failed, while the second entry shows a successful `su` command.

The `sudo` example shows a successful command using `sudo`. The entry includes:

- Terminal window
- Current working directory
- Escalated user account
- Command



User account creation can also be of immense interest to an investigator. Audit records for user creation are very verbose. Lots of related entries are created. Audit records also have more detail than records found in the `system.log` or `unified logs`.

One of the records, shown in the example above, contains the event "`create user`". This entry includes the following data:

- Timestamp
- User who created the new user (uid=501)
- Name of new user (`supersecretuser`)

In the next example, the `system.log/unified` entry shows who unlocked the account preferences pane. In this log event, it does not show if a new user account was created or what the account name was. In fact, by just looking at this log entry we wouldn't know a new account was created.

This is a good example of using multiple sources to get a more detailed picture.

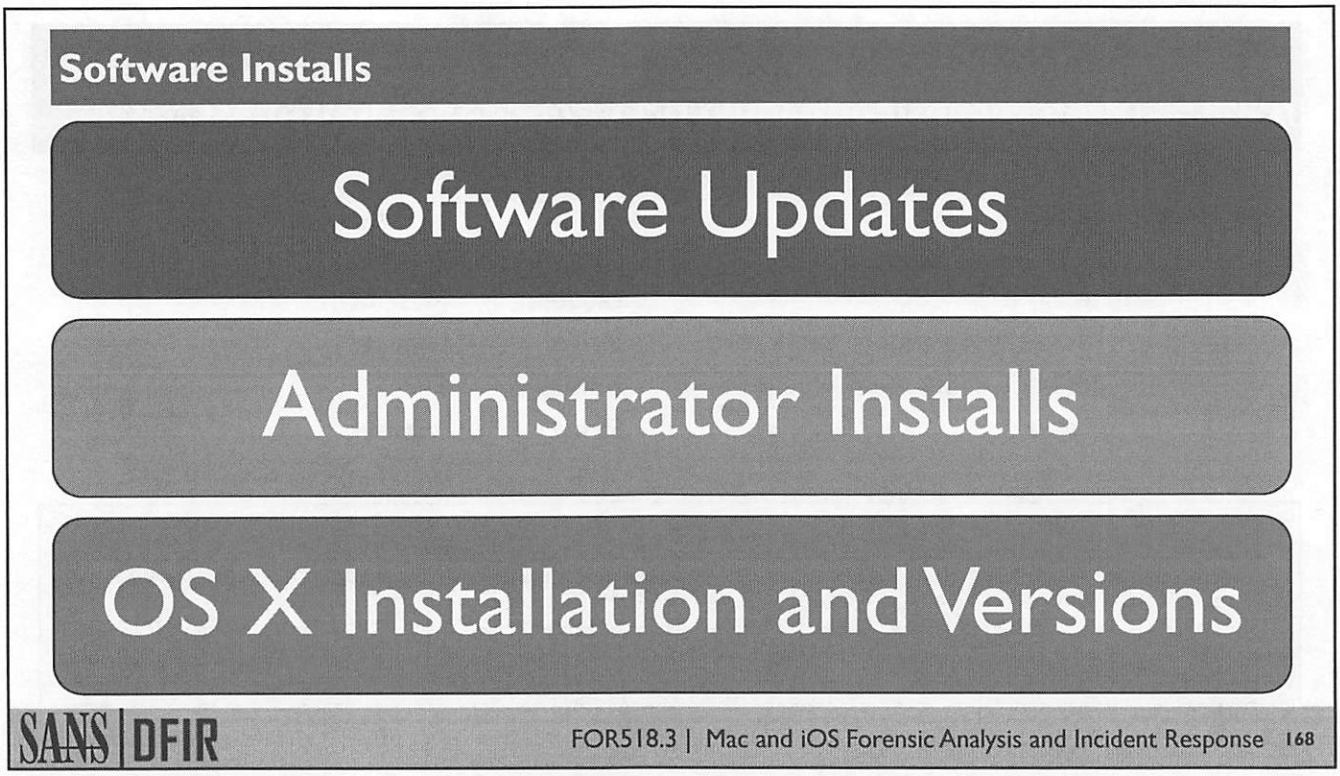
Account Deletion
/Library/Preferences/com.apple.preferences.accounts.plist

Key	Type	Value
▼ deletedUsers	Array	(2 items)
▶ Item 0	Diction...	(4 items)
▼ Item 1	Diction...	(4 items)
dsAttrTypeStandard:RealName	String	testuser
dsAttrTypeStandard:UniqueID	Number	502
name	String	testuser
date	Date	Jun 13, 2012 8:41:58 PM

```
<record version="11" event="delete user" modifier="0" time="Wed Jun 13 20:41:56 2012" msec="
+ 322 msec" >
<subject audit-uid="501" uid="501" gid="20" ruid="501" rgid="20" pid="10717" sid="100005"
tid="10717 0.0.0.0" />
<text>Delete record type Users &apos;testuser&apos; node &apos;/Local/Default&apos;</text>
<return errval="success" retval="0" />
</record>
```


If account creation is important, then so is account deletion. The same data can be found in the audit logs, as can be found for account creation. The event "`delete user`" records the user (501) who deleted the account, what account was deleted (`testuser`), and when.

The deleted users are shown in the `com.apple.preferences.accounts.plist` file under the `deletedUsers` key. This key contains the deleted user's name, UID, username, and the deletion date in local system time.



Logs can show us when the software was installed and if Administrator privileges were needed. These logs can also show what OS X version is installed and when it was updated.

Install Details: /var/log/install.log

```
May 27 11:59:03 MBP Installer[470]: logKext  Installation Log
May 27 11:59:03 MBP Installer[470]: Opened from:
/Users/oompa/Downloads/logKext-2.3.pkg
May 27 11:59:03 MBP Installer[470]: Product archive
/Users/oompa/Downloads/logKext-2.3.pkg trustLevel=100
May 27 11:59:17 MBP Installer[470]: InstallerStatusNotifications plugin
loaded
May 27 11:59:26 MBP runner[477]: Administrator authorization granted.
May 27 11:59:26 MBP Installer[470]:
================================================================================
=======
May 27 11:59:26 MBP Installer[470]: User picked Standard Install
May 27 11:59:26 MBP Installer[470]: Choices selected for installation:
...
May 27 12:01:34 MBP installd[481]: Installed "logKext" ()
May 27 12:01:35 MBP installd[481]: PackageKit: ----- End install -----
```


The `install.log` also contains software install details, such as:

- Where the software package was opened from on disk
- Was administrator authorization needed?

```
Jun  9 12:43:51 nibble.blah installd[64413]: PackageKit: ----- Begin install -----
Jun  9 12:43:51 nibble.blah installd[64413]: PackageKit: request=PKInstallRequest <1 packages, destination=/>
Jun  9 12:43:51 nibble.blah installd[64413]: PackageKit: packages=(
                "PKLeopardPackage <file://localhost/var/folders/f1/_wpdftvx3k3_c96vhxd0fkqr0000gn/C/com.apple.appstore/404458553/
mzps1501882897890740536.pkg#com.omnigroup.OmniGraffle.MacAppStore.pkg>"
            )
Jun  9 12:43:52 nibble.blah installd[64413]: PackageKit: Extracting file://localhost/var/folders/f1/_wpdftvx3k3_c96vhxd0fkqr0000gn/C/
com.apple.appstore/404458553/mzps1501882897890740536.pkg#com.omnigroup.OmniGraffle.MacAppStore.pkg (destination=/var/folders/zz/
zyxvpxvq6csfxvn_n0000000000000/Cleanup At Startup/PKInstallSandboxManager/3.sandbox/Root/Applications, uid=0)
Jun  9 12:43:53 nibble.blah installd[64413]: PackageKit: Applying atomic-update from bundle at Applications/OmniGraffle 5.app
Jun  9 12:43:56 nibble.blah Software Update[71745]: PackageKit: Missing bundle path, skipping: <bundle id="com.apple.SystemProfiler"></
bundle>
Jun  9 12:43:56 nibble.blah Software Update[71745]: PackageKit: Missing bundle path, skipping: <bundle
id="com.apple.java.JavaPreferences"></bundle>
Jun  9 12:43:56 nibble.blah Software Update[71745]: PackageKit: Missing bundle path, skipping: <bundle id="com.apple.iCal"></bundle>
Jun  9 12:43:58 nibble.blah installd[64413]: PackageKit: Verifying code signature on /var/folders/zz/zyxvpxvq6csfxvn_n0000000000000/Cleanup
At Startup/PKInstallSandboxManager/3.sandbox/Root/Applications/OmniGraffle 5.app
Jun  9 12:44:01 nibble.blah installd[64413]: PackageKit: Wrote MAS receipt into Applications/OmniGraffle 5.app
Jun  9 12:44:01 nibble.blah installd[64413]: PackageKit: prevent user idle system sleep
Jun  9 12:44:01 nibble.blah installd[64413]: PackageKit: suspending backupd
Jun  9 12:44:01 nibble.blah install_monitor[71764]: Temporarily excluding: /Applications, /Library, /System, /bin, /private, /sbin, /usr
Jun  9 12:44:03 nibble.blah installd[64413]: PackageKit: Shoving /var/folders/zz/zyxvpxvq6csfxvn_n0000000000000/Cleanup At Startup/
PKInstallSandboxManager/3.sandbox/Root (1 items) to /
Jun  9 12:44:03 nibble.blah installd[64413]: PackageKit: Writing receipt for com.omnigroup.OmniGraffle.MacAppStore to /private/var/db/
receipts
Jun  9 12:44:03 nibble.blah installd[64413]: PackageKit: Touched bundle Applications/OmniGraffle 5.app
Jun  9 12:44:03 nibble.blah installd[64413]: PackageKit: Touched bundle Applications/OmniGraffle 5.app/Contents/Resources/
OmniGroupCrashCatcher.app
Jun  9 12:44:03 nibble.blah installd[64413]: Installed "OmniGraffle" (5.4.3)
```

Installed Software: /var/log/install.log—Search "Installed"

```
May  9 16:28:06 localhost OSInstaller[328]: Installed "Mac OS X" ()
…
May  9 19:56:21 bit installd[338]: Installed "Evernote" ()
May 10 00:45:34 bit installd[559]: Installed "Flashback malware removal tool" (1.0)
May 10 00:45:34 bit installd[559]: Installed "Mac OS X Update Combined" (10.7.4)
May 10 00:45:34 bit installd[559]: Installed "iTunes" (10.6.1)
May 10 00:46:33 bit installd[559]: Installed "Lion Recovery Update" (1.0)
May 10 16:51:51 bit installd[295]: Installed "Xcode" ()
May 10 16:55:55 bit installd[295]: Installed "iPhoto" ()
May 11 19:51:09 bit installd[4384]: Installed "Office 2011 14.1.0 Update" ()
May 14 18:31:44 bit installd[9572]: Installed "Java for OS X 2012-003" (1.0)
May 19 16:50:20 bit installd[20691]: Installed "TrueCrypt 7.1a" ()
May 19 17:17:25 bit installd[20847]: Installed "CCleaner" ()
May 19 17:32:19 bit installd[20847]: Installed "TextWrangler" ()
May 26 20:15:45 bit installd[39022]: Installed "The Unarchiver" ()
May 27 15:46:56 bit installd[41936]: Installed "Wireshark 1.6.8 Intel 64" ()
May 27 20:57:48 bit installd[514]: Installed "Microsoft Error Reporting for Mac" ()
May 27 20:59:41 bit installd[978]: Installed "Office 2011 14.2.2 Update" ()
```


Software installations can be a good resource to find out what types of software are used on the system and when they were downloaded.

We can look at the `install.log` on the system and search for the keyword "`Installed`" (or more specifically "`: Installed`") to easily show the software packages installed. This list will not only include user-initiated software like TrueCrypt and TextWrangler but also system software updates like the iTunes 10.6.1 update.

This log does not include installation information of command-line tools such as `macports` or `fink` repositories.

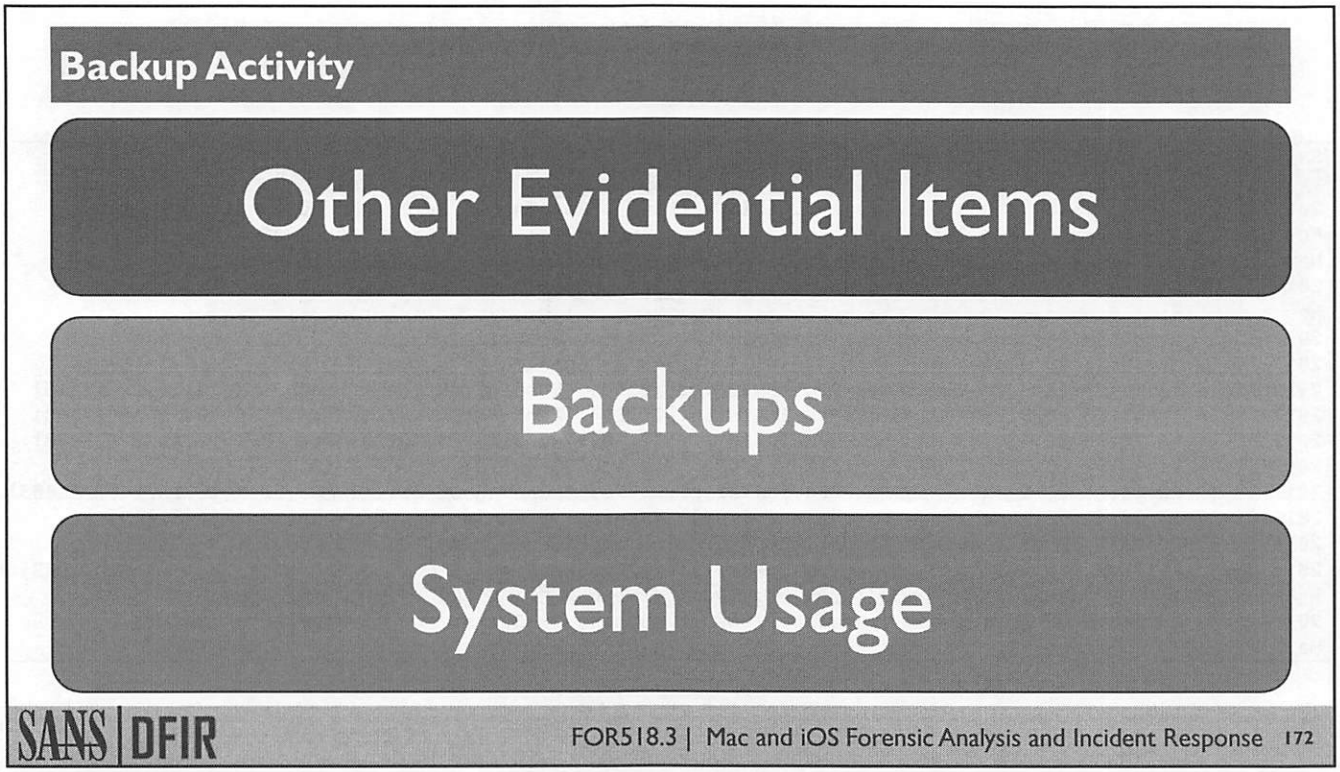
Caveat: Software installed via a "Drag and Drop" method such as Firefox and Chrome will not show in this log. Some software allows a user to copy the application directly to the `/Application` directory.

System Version: /var/log/install.log—Search "Running OS Build:"

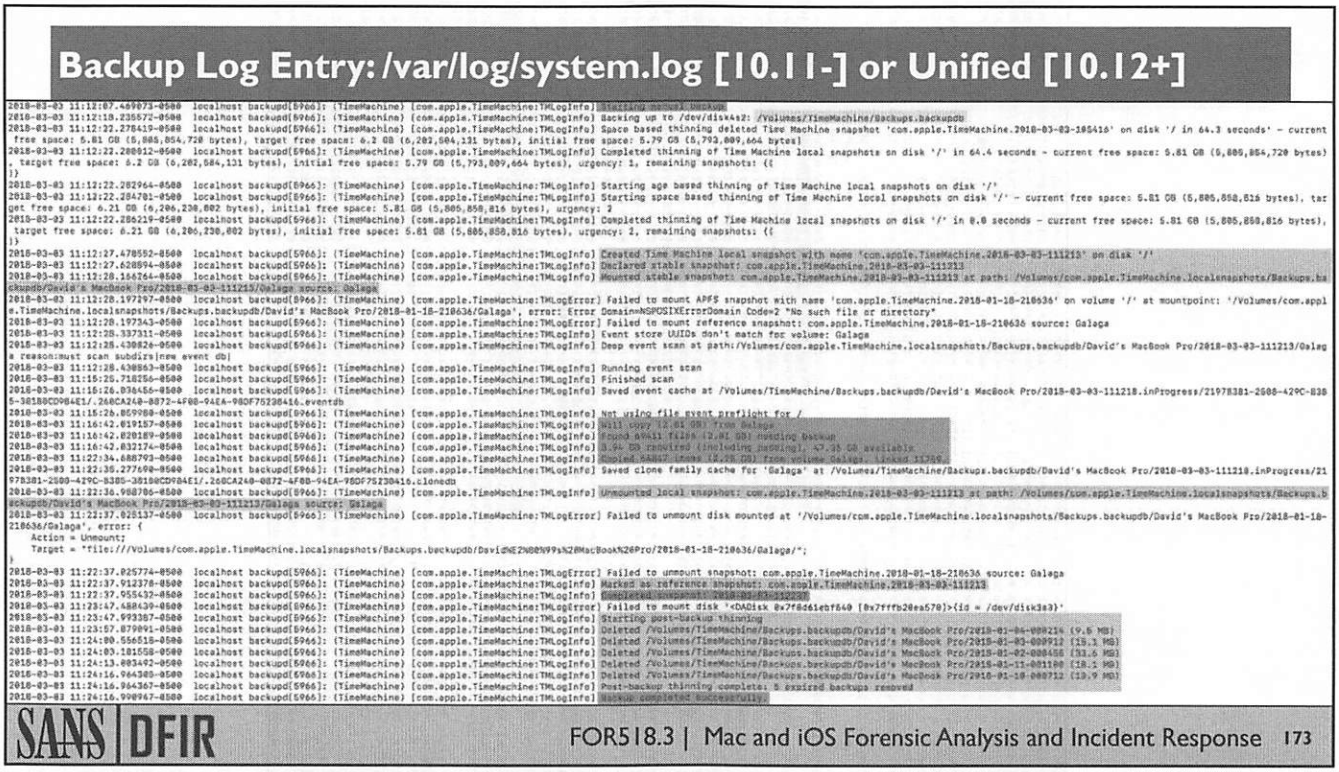
```
Sarahs-MBP-5:~ oompa$ less /var/log/install.log | grep "Running OS Build"
2017-11-11 17:53:40+00 MacBook-Pro InstallAssistant[556]: Running OS Build: Mac OS X 10.13 (17A365)
2017-11-11 17:56:32+00 MacBook-Pro InstallAssistant[609]: Running OS Build: Mac OS X 10.13 (17A365)
Nov 11 18:01:31 MacBook-Pro OSInstaller[550]: Running OS Build: Mac OS X 10.13 (17A365)
Nov 11 18:05:45 MacBook-Pro OSInstaller[550]: Running OS Build: Mac OS X 10.13 (17A365)
2017-11-11 14:47:49-05 MacBook-Pro Installer[2933]: Running OS Build: Mac OS X 10.13 (17A365)
Nov 11 20:53:47 MacBook-Pro OSInstaller[558]: Running OS Build: Mac OS X 10.13.1 (17B48)
2017-11-11 16:56:00-05 MacBook-Pro Installer[1505]: Running OS Build: Mac OS X 10.13.1 (17B48)
2017-11-12 22:59:21-05 Sarahs-MBP InstallAssistant[2486]: Running OS Build: Mac OS X 10.13.1 (17B48)
2017-11-16 14:11:37-08 Sarahs-MacBook-Pro Installer[19193]: Running OS Build: Mac OS X 10.13.1 (17B48)
2017-11-16 14:13:23-08 Sarahs-MacBook-Pro Installer[19234]: Running OS Build: Mac OS X 10.13.1 (17B48)
2017-11-16 14:20:39-08 Sarahs-MacBook-Pro Installer[37128]: Running OS Build: Mac OS X 10.13.1 (17B48)
2017-11-25 22:30:07-05 Sarahs-MBP Installer[3793]: Running OS Build: Mac OS X 10.13.1 (17B48)
2017-12-17 22:06:15-05 Sarahs-MacBook-Pro Installer[23976]: Running OS Build: Mac OS X 10.13.1 (17B1003)
2018-01-19 13:51:41-05 sarahs-mbp Installer[48509]: Running OS Build: Mac OS X 10.13.1 (17B1003)
2018-01-23 20:15:26-05 Sarahs-MBP Installer[659]: Running OS Build: Mac OS X 10.13.1 (17B1003)
2018-02-10 11:29:46+00 Sarahs-MacBook-Pro Installer[2280]: Running OS Build: Mac OS X 10.13.1 (17B1003)
Feb 17 13:46:08 MacBook-Pro OSInstaller[550]: Running OS Build: Mac OS X 10.13.3 (17D47)
2018-03-03 20:24:16-05 Sarahs-MBP Installer[74593]: Running OS Build: Mac OS X 10.13.3 (17D47)
Mar  7 02:04:19 MacBook-Pro OSInstaller[561]: Running OS Build: Mac OS X 10.13.3 (17D47)
```


A historical view of the system version can be seen if a search for "Running OS `Build:`" is performed.

The example above shows a system being upgraded from 10.13 to 10.13.1 to 10.13.3.



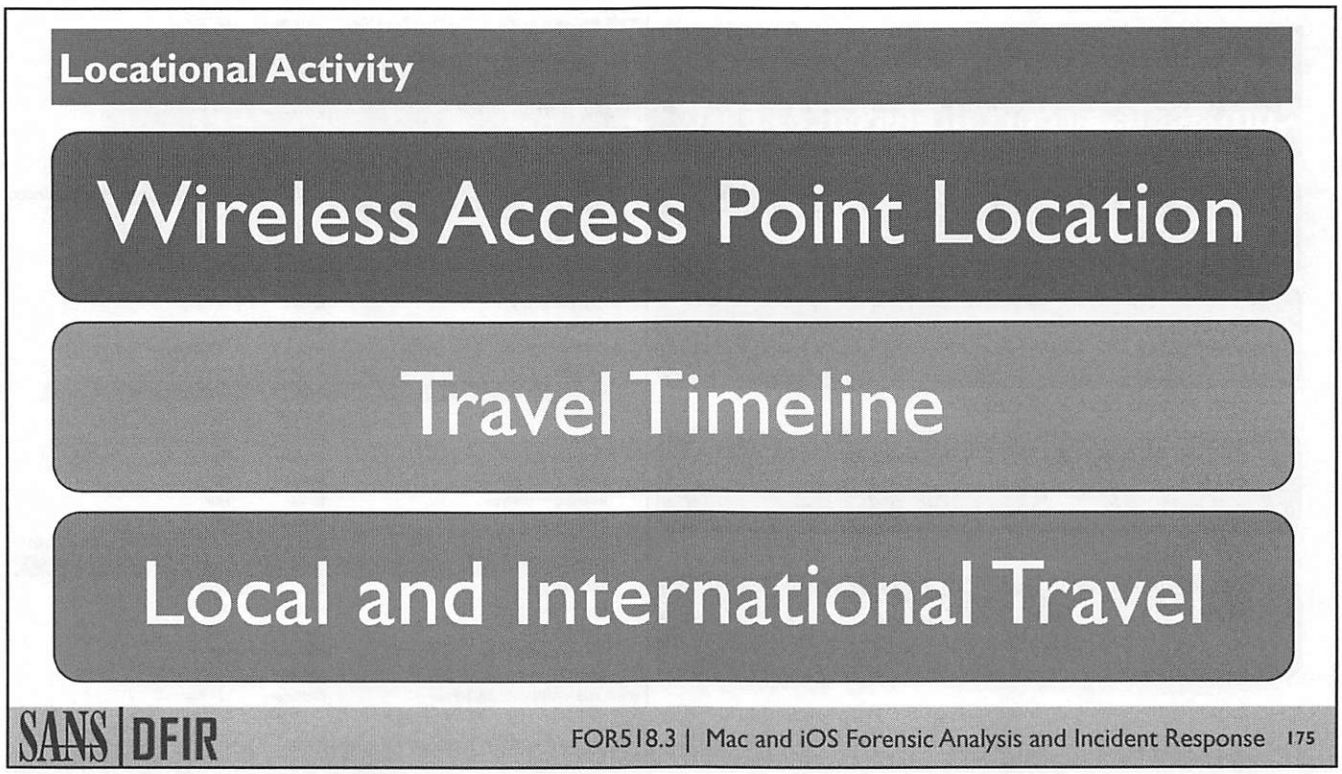
Evidence of backups can provide new or unknown evidential items for investigators. Backups can provide snapshots in time of a particular system and can show how often, as well as how much, a system was used.



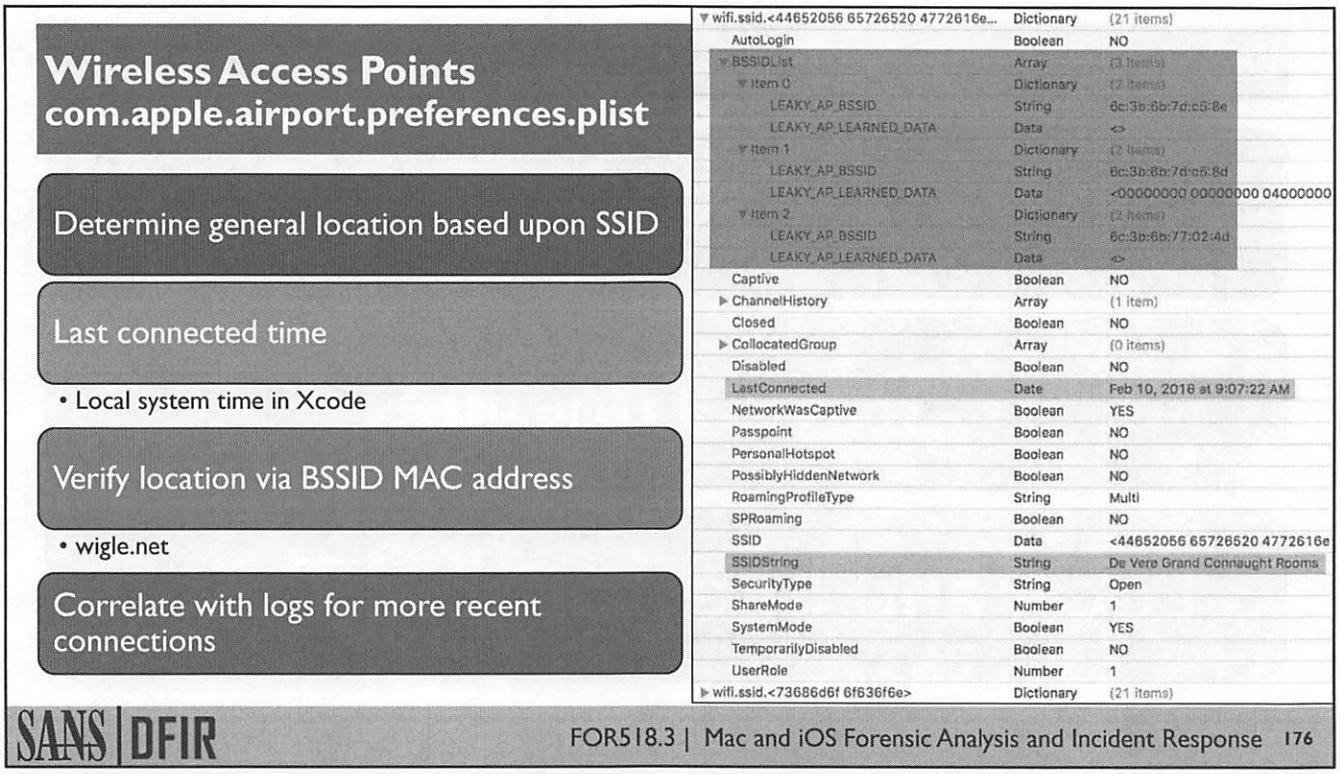
Access to other systems that contain backups could be a potential source of more evidence. OS X uses the application Time Machine to interact with backup drives. The `system.log` file contains various backup-related events, such as:

- When the backup was started
- Network location of backup (or local location if a USB HDD is used)
- Local mount point of network hard drive
- Amount of data backed up
- Volume name to be backed up
- Deletion of old backups
- When the backup was completed

```
2018-03-03 11:12:07.469073-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Starting manual backup
2018-03-03 11:12:18.235572-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Backing up to /dev/disk4s2: /Volumes/TimeMachine/Backups.backupdb
2018-03-03 11:12:22.278419-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Space based thinning deleted Time Machine snapshot 'com.apple.TimeMachine.2018-03-03-105416' on disk '/ in 64.3 seconds' - current
 free space: 5.81 GB (5,805,854,720 bytes), target free space: 6.2 GB (6,202,504,131 bytes), initial free space: 5.79 GB (5,793,009,664 bytes)
2018-03-03 11:12:22.280012-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Completed thinning of Time Machine local snapshots on disk '/' in 64.4 seconds - current free space: 5.81 GB (5,805,854,720 bytes)
, target free space: 6.2 GB (6,202,504,131 bytes), initial free space: 5.79 GB (5,793,009,664 bytes), urgency: 1, remaining snapshots: {{
}}
2018-03-03 11:12:22.282964-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Starting age based thinning of Time Machine local snapshots on disk '/'
2018-03-03 11:12:22.284701-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Starting space based thinning of Time Machine local snapshots on disk '/' - current free space: 5.81 GB (5,805,858,816 bytes), tar
get free space: 6.21 GB (6,206,230,002 bytes), initial free space: 5.81 GB (5,805,858,816 bytes), urgency: 2
2018-03-03 11:12:22.286219-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Completed thinning of Time Machine local snapshots on disk '/' in 0.0 seconds - current free space: 5.81 GB (5,805,858,816 bytes),
 target free space: 6.21 GB (6,206,230,002 bytes), initial free space: 5.81 GB (5,805,858,816 bytes), urgency: 2, remaining snapshots: {{
}}
2018-03-03 11:12:27.470552-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Created Time Machine local snapshot with name 'com.apple.TimeMachine.2018-03-03-111213' on disk '/'
2018-03-03 11:12:27.620894-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Declared stable snapshot: com.apple.TimeMachine.2018-03-03-111213
2018-03-03 11:12:28.166254-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Mounted stable snapshot: com.apple.TimeMachine.2018-03-03-111213 at path: /Volumes/com.apple.TimeMachine.localsnapshots/Backups.ba
ckupdb/David's MacBook Pro/2018-03-03-111213/Galaga source: Galaga
2018-03-03 11:12:28.197297-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed to mount APFS snapshot with name 'com.apple.TimeMachine.2018-01-18-210636' on volume '/' at mountpoint: '/Volumes/com.appl
e.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-01-18-210636/Galaga', error: Error Domain=NSPOSIXErrorDomain Code=2 "No such file or directory"
2018-03-03 11:12:28.197343-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed to mount reference snapshot: com.apple.TimeMachine.2018-01-18-210636 source: Galaga
2018-03-03 11:12:28.337311-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Event store UUIDs don't match for volume: Galaga
2018-03-03 11:12:28.430826-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Deep event scan at path:/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-03-03-111213/Galag
a reason:must scan subdirs|new event db|
2018-03-03 11:12:28.430863-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Running event scan
2018-03-03 11:15:25.718256-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Finished scan
2018-03-03 11:15:26.036456-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Saved event cache at /Volumes/TimeMachine/Backups.backupdb/David's MacBook Pro/2018-03-03-111218.inProgress/2197B381-2508-429C-B38
5-381B0CD984E1/.268CA240-0872-4F08-94EA-98DF75238416.eventdb
2018-03-03 11:15:26.059980-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Not using file event preflight for /
2018-03-03 11:16:42.019157-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Will copy (2.81 GB) from Galaga
2018-03-03 11:16:42.020189-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Found 69411 files (2.81 GB) needing backup
2018-03-03 11:16:42.032174-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] 3.94 GB required (including padding), 47.35 GB available
2018-03-03 11:22:34.688793-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Copied 66847 items (2.25 GB) from volume Galaga. Linked 11759.
2018-03-03 11:22:35.277690-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Saved clone family cache for 'Galaga' at /Volumes/TimeMachine/Backups.backupdb/David's MacBook Pro/2018-03-03-111218.inProgress/21
97B381-2508-429C-B385-381B0CD984E1/.268CA240-0872-4F08-94EA-98DF75238416.clonedb
2018-03-03 11:22:36.988706-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Unmounted local snapshot: com.apple.TimeMachine.2018-03-03-111213 at path: /Volumes/com.apple.TimeMachine.localsnapshots/Backups.b
ackupdb/David's MacBook Pro/2018-03-03-111213/Galaga source: Galaga
2018-03-03 11:22:37.025137-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-01-18-
210636/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-01-18-210636/Galaga/";
}
2018-03-03 11:22:37.025774-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed to unmount snapshot: com.apple.TimeMachine.2018-01-18-210636 source: Galaga
2018-03-03 11:22:37.912378-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Marked as reference snapshot: com.apple.TimeMachine.2018-03-03-111213
2018-03-03 11:22:37.955432-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Completed snapshot: 2018-03-03-112237
2018-03-03 11:23:47.488439-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed to mount disk '<DADisk 0x7f8d61ebf540 [0x7fffb20ea570]>{id = /dev/disk3s3}'
2018-03-03 11:23:47.993387-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Starting post-backup thinning
2018-03-03 11:23:57.079091-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Deleted /Volumes/TimeMachine/Backups.backupdb/David's MacBook Pro/2018-01-04-000214 (9.5 MB)
2018-03-03 11:24:00.556518-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Deleted /Volumes/TimeMachine/Backups.backupdb/David's MacBook Pro/2018-01-03-000912 (15.1 MB)
2018-03-03 11:24:03.101558-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Deleted /Volumes/TimeMachine/Backups.backupdb/David's MacBook Pro/2018-01-02-000455 (33.6 MB)
2018-03-03 11:24:13.003492-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Deleted /Volumes/TimeMachine/Backups.backupdb/David's MacBook Pro/2018-01-11-001100 (18.1 MB)
2018-03-03 11:24:16.964305-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Deleted /Volumes/TimeMachine/Backups.backupdb/David's MacBook Pro/2018-01-10-000712 (13.9 MB)
2018-03-03 11:24:16.964367-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Post-backup thinning complete: 5 expired backups removed
2018-03-03 11:24:16.990947-0500  localhost backupd[5966]: (TimeMachine) [com.apple.TimeMachine:TMLogInfo] Backup completed successfully.
```



There are many data points in logs showing locational activity, specifically on laptop systems. Laptops are meant to be mobile and to travel around the city, country, or world. The logs can provide a timeline of travel that can help an investigator correlate where a user may have been at a particular time.



The `com.apple.airport.preferences.plist` file contains the "known" networks. These networks are saved by default on macOS until the user purges them.

This plist is a good start to determine where a user might have been by the access points they connected to. Note that if this plist is viewed within Xcode, these "Last Connected" timestamps are in local system time to the analysis system.

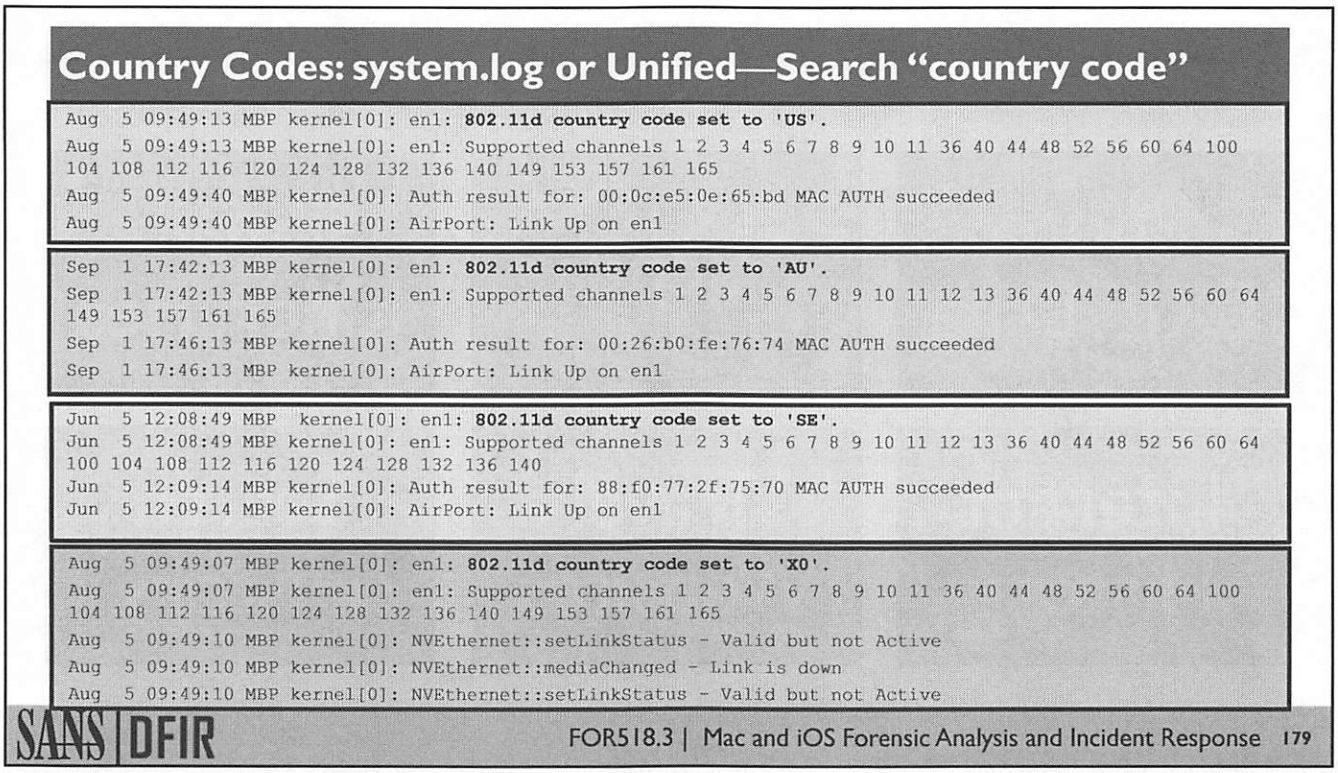
The BSSID MAC address can be searched for in services like wigle.net to determine the potential physical location of these networks.

To get a more granular timeline, correlate these connections in the logs.

Detailed Timeline: system.log/Unified Logs Search "configd" or "BSSID" or "en0"

Searching for "`configd`"/"BSSID"/"en0" in the `system.log/Unified logs` can show a more detailed timeline of wireless activity. The same network access points mentioned previously are connected at specific times.

```
2018-02-10 14:06:53.910380+0000  localhost configd[53]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID De Vere Grand Connaught Rooms BSSID 6c:3b:6b:7d:c5:8d
2018-02-10 14:07:21.699310+0000  localhost configd[53]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID De Vere Grand Connaught Rooms BSSID 6c:3b:6b:7d:c5:8d
2018-02-10 14:07:21.704505+0000  localhost configd[53]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID De Vere Grand Connaught Rooms BSSID 6c:3b:6b:7d:c5:8d
2018-02-10 14:07:22.740626+0000  localhost configd[53]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID De Vere Grand Connaught Rooms BSSID 6c:3b:6b:7d:c5:8d
2018-02-25 19:16:41.475644+0000  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 19:16:41.479578+0000  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 19:16:41.713504+0000  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 19:16:41.744678+0000  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 15:51:35.603724-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:35.608307-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:35.687538-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:35.698212-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:39.076707-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:39.080212-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:39.174335-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:39.262729-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:22.594247-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:23.304215-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:31.051236-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:31.056675-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:31.118892-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:31.194112-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:56:21.687936-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 15:56:21.692137-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:29.166817-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:18:29.171309-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:18:29.478367-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:18:29.558086-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:18:53.088836-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:53.096574-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:56.454316-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:56.459972-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:59.949000-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:18:59.966975-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:19:00.186917-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:19:00.263482-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:19:12.790885-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:19:12.796031-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:19:13.777743-0500  localhost configd[54]: (IPConfiguration) [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
```

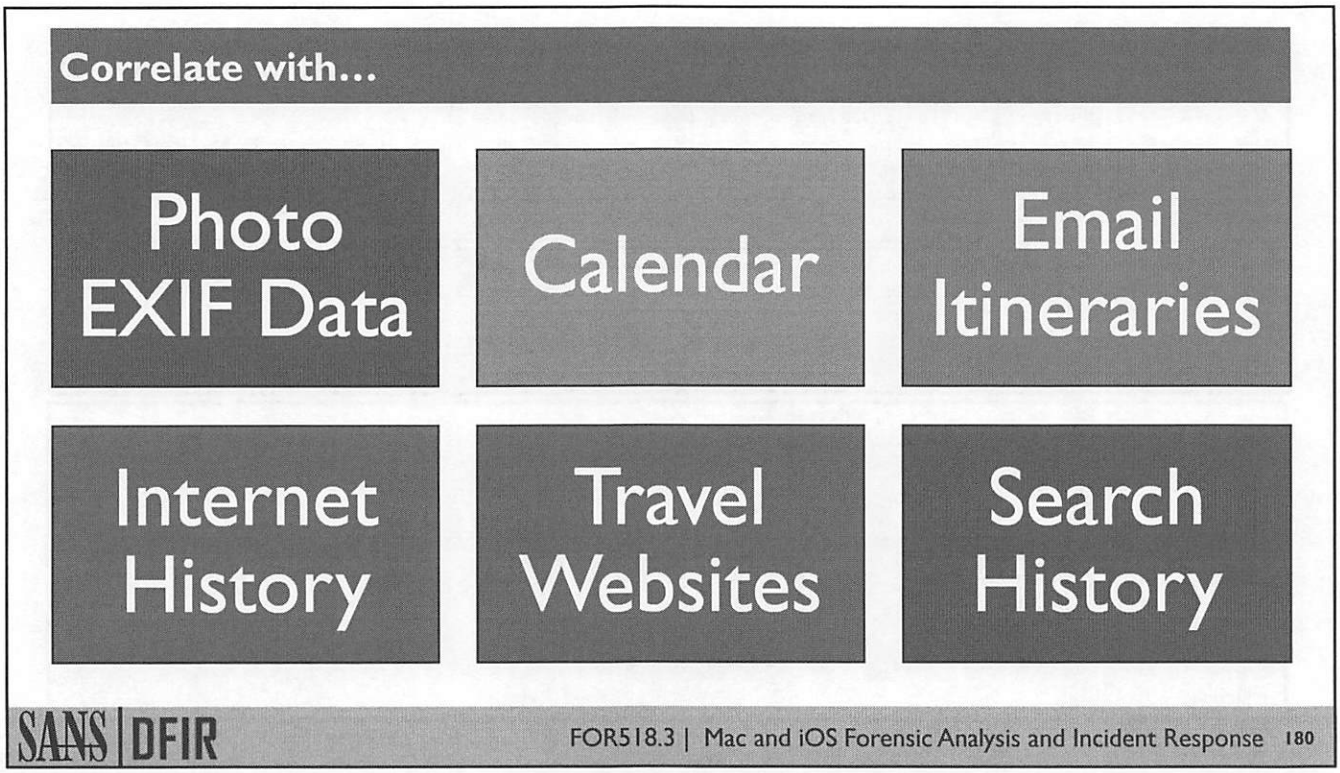


International travel may also be very interesting for your investigation. The country codes for wireless access points are recorded in the `kernel.log` (in 10.7 and before) the `system.log` (10.8+), and Unified Logs in 10.12+.

802.11d is an amendment to 802.11 that allows specification on regulatory domains to include country information being included by beacons.

In the example above, the country codes are shown whenever a connection is made to a wireless access point. Each connection is displayed in a different color to show related records.

- The first example shows a connection to an access point in the United States (`US`) on August 5.
- The second example shows it was connected to an access point in Australia (`AU`) on September 1.
- The third example on June 5 shows a connection to a wireless AP in Sweden (`SE`).
- The fourth example to the country code (`X0`) is the default country code when one is not available, or is being determined.



The travel data found in the logs can be correlated with many other forensic artifacts.

Lab 3.4
Timeline Analysis and Data Correlation

This page intentionally left blank.

FOR518.3 | Mac and iOS Forensic Analysis and Incident Response

FOR518 Section 3: System Configuration and Log Analysis

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

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

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

SANS Programs
sans.org/programs

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

Search SANSInstitute

SANS Free Resources
sans.org/security-resources

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

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