SANS

www.sans.org

FORENSICS 518

MAC FORENSIC ANALYSIS

518.3

System and Local Domain File Analysis

The right security training for your staff, at the right time, in the right location.

IMPORTANT-READ CAREFULLY:

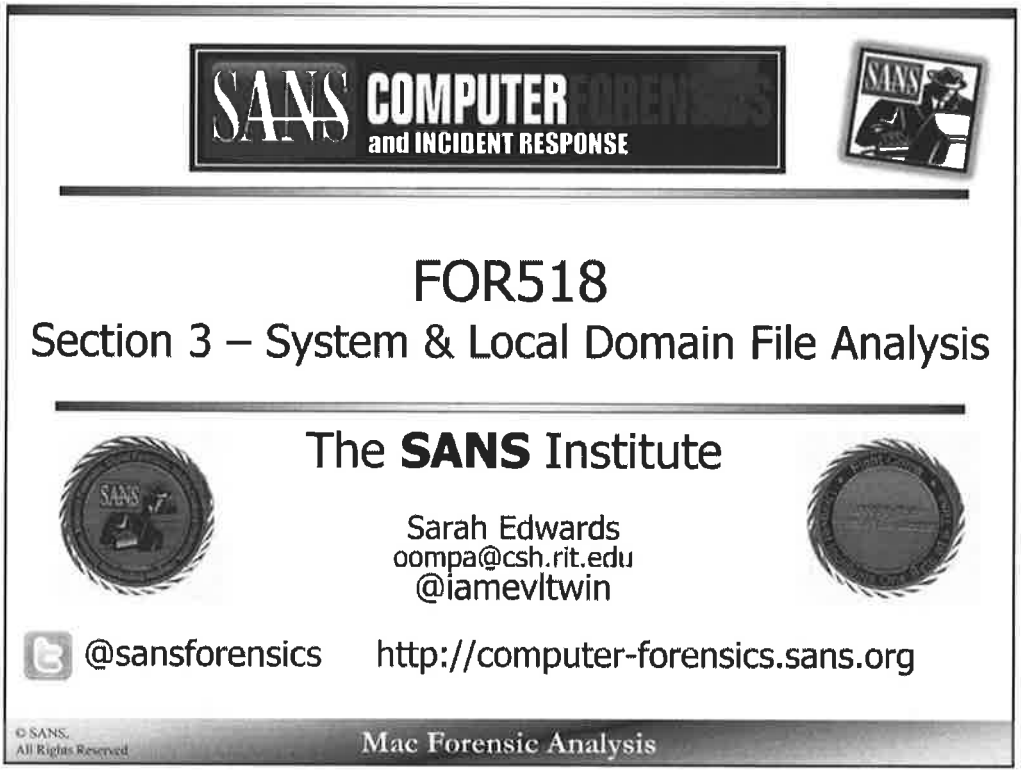
This Courseware License Agreement ("CLA") is a legal agreement between you (either an individual or a single entity; henceforth User) and the SANS Institute for the personal, non-transferable use of this courseware. User agrees that the CLA is the complete and exclusive statement of agreement between The SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA. If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this courseware. BY ACCEPTING THIS COURSEWARE YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. IF YOU DO NOT AGREE YOU MAY RETURN IT TO THE SANS INSTITUTE FOR A FULL REFUND, IF APPLICABLE. The SANS Institute hereby grants User a non-exclusive license to use the material contained in this courseware subject to the terms of this agreement. User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of this publication in any medium whether printed, electronic or otherwise, for any purpose without the express written consent of the SANS Institute. Additionally, user may not sell, rent, lease, trade, or otherwise transfer the courseware in any way, shape, or form without the express written consent of the SANS Institute.

The SANS Institute reserves the right to terminate the above lease at any time. Upon termination of the lease, user is obligated to return all materials covered by the lease within a reasonable amount of time.

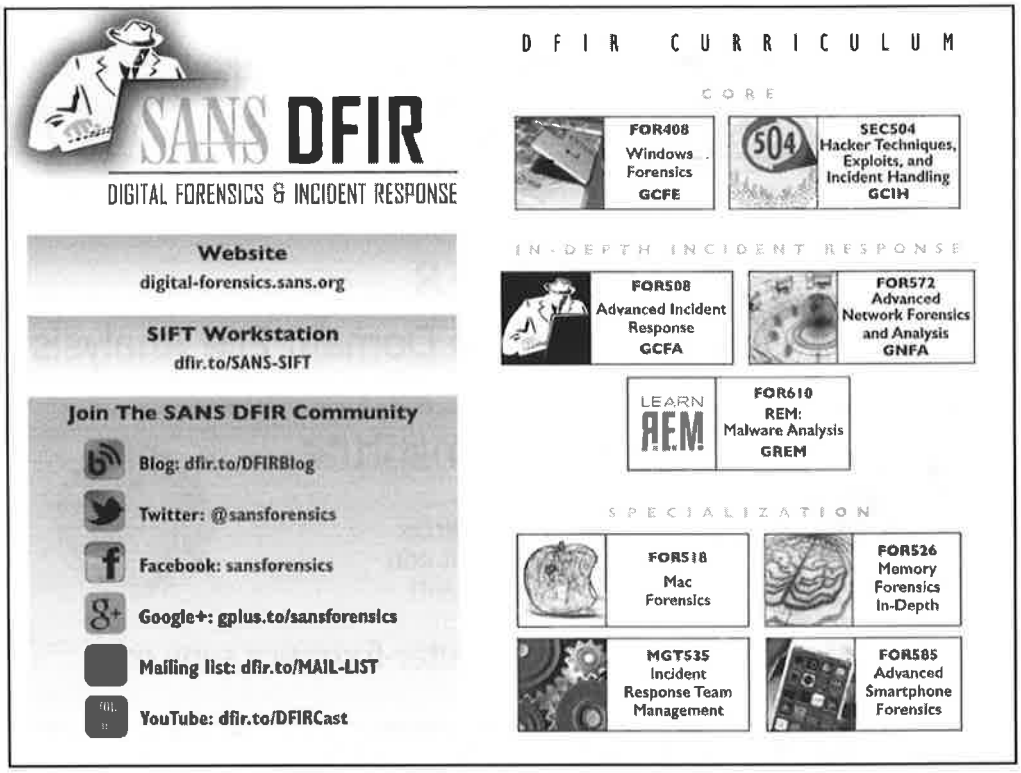
SANS acknowledges that any and all software and/or tools presented in this courseware are the sole property of their respective trademark/registered/copyright owners.

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

For518_3_A05_01



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



This page intentionally left blank.

SANS DFIR

DIGITAL FORENSICS & INCIDENT RESPONSE

Website
digital-forensics.sans.org

SIFT Workstation
dfir.to/SANS-SIFT

Join The SANS DFIR Community

- Blog: dfir.to/DFIRBlog
- Twitter: @sansforensics
- Facebook: sansforensics
- Google+: gplus.to/sansforensics
- Mailing list: dfir.to/MAIL-LIST
- YouTube: dfir.to/DFIRCast

DFIR CURRICULUM

CORE

FOR408
Windows Forensics
GCFE

SEC504
Hacker Techniques, Exploits, and Incident Handling
GCIH

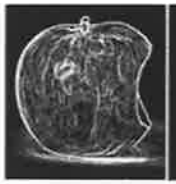
IN-DEPTH INCIDENT RESPONSE

FOR508
Advanced Incident Response
GCFA

FOR572
Advanced Network Forensics and Analysis
GNFA

FOR610
REM: Malware Analysis
GREM

SPECIALIZATION

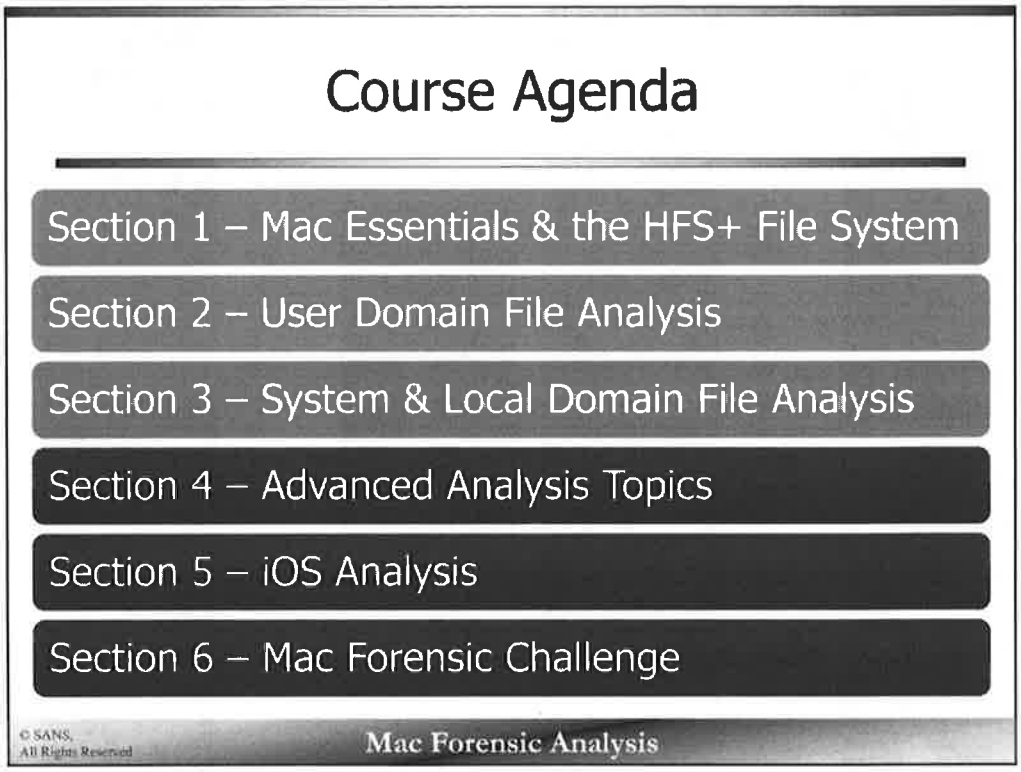


FOR518
Mac Forensics

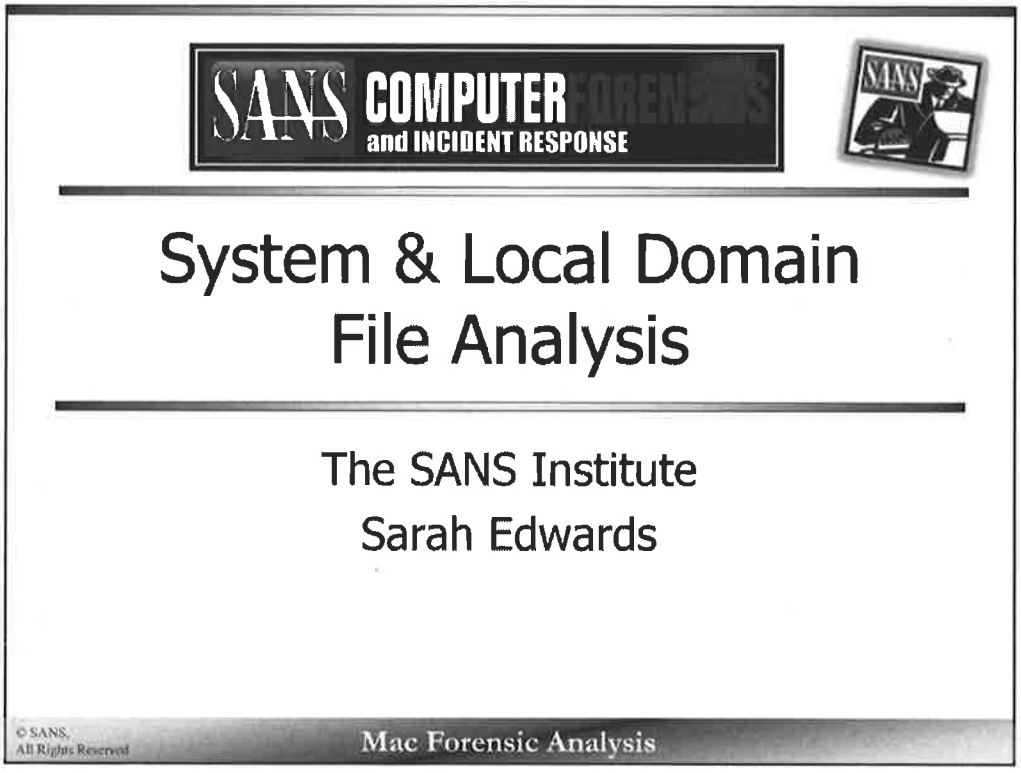
FOR526
Memory Forensics In-Depth

MGT535
Incident Response Team Management

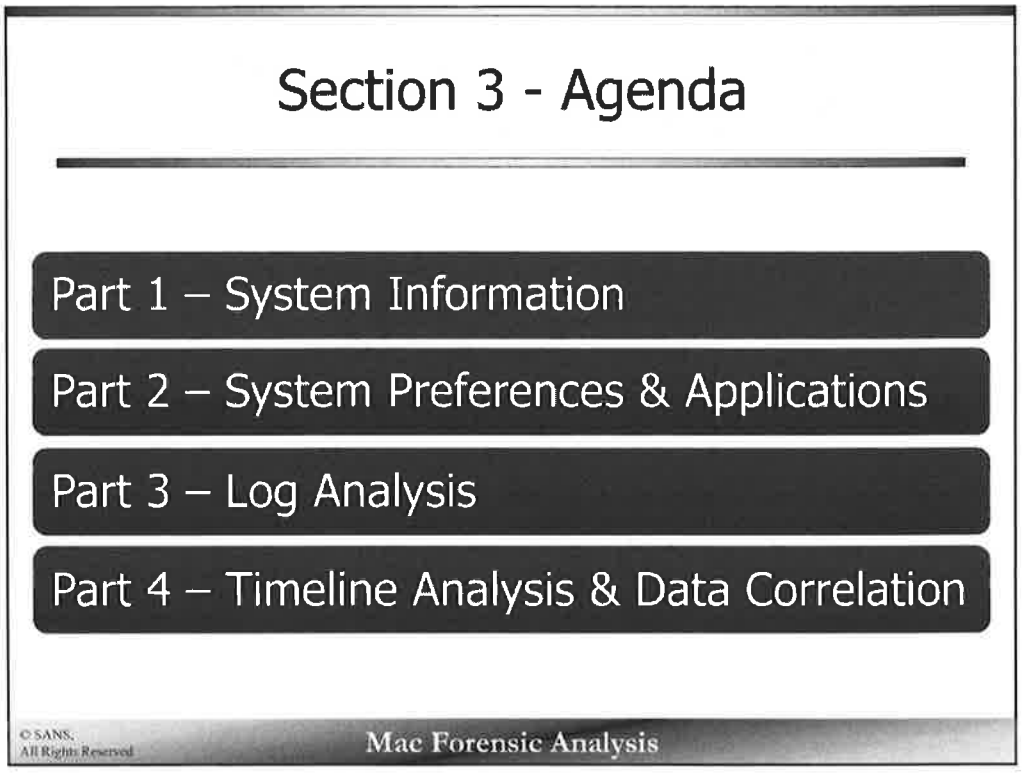
FOR585
Advanced Smartphone Forensics



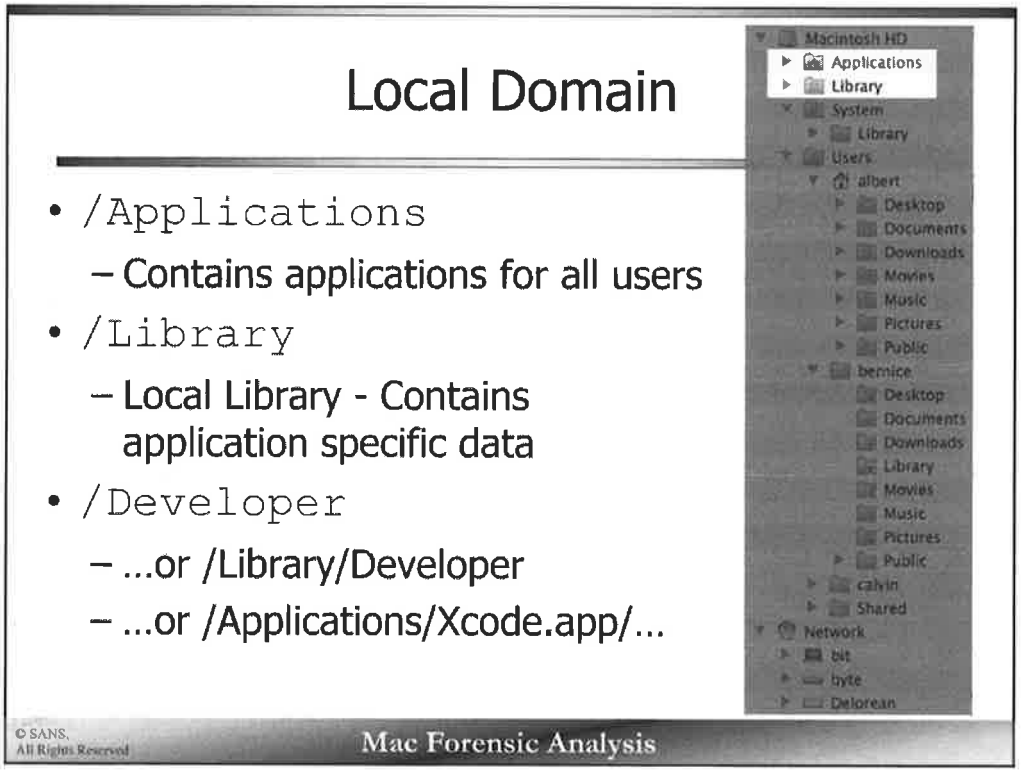
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.



The local domain consists of the `Applications` directory, the (local) `Library` directory, and if installed, the `Developer` directory. This domain is used to store the files that may be shared amongst the users, such as applications.

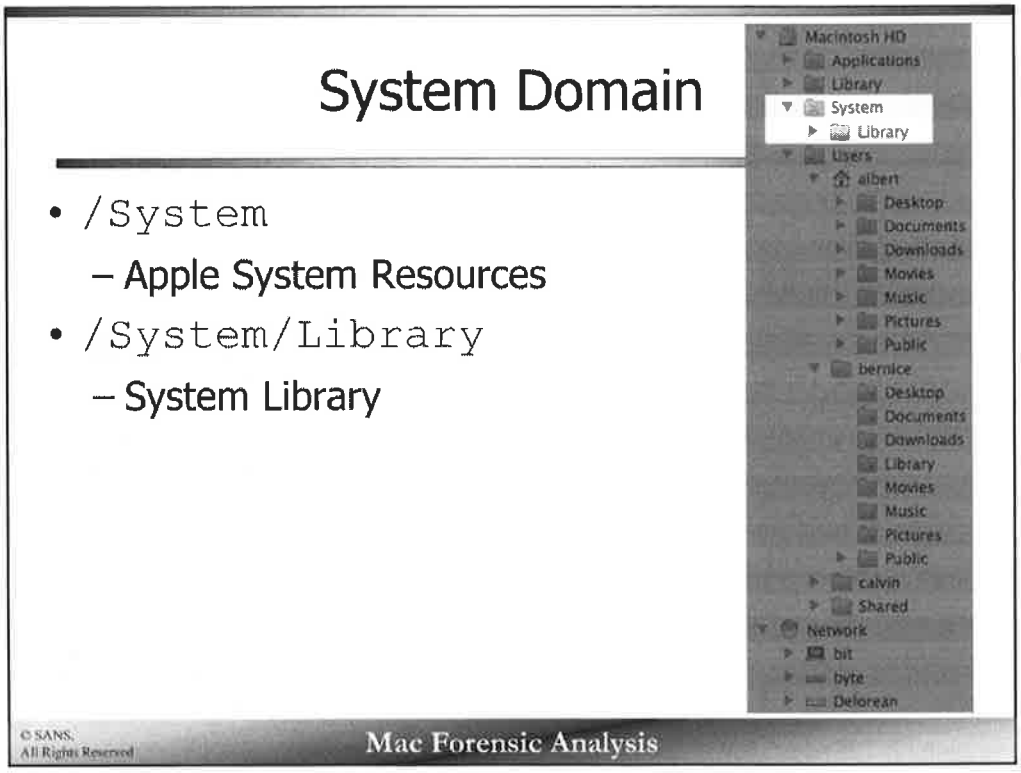
The `/Applications` directory contains the programs available to all users of the system. It is not necessary to run an application from the `/Applications` directory. It may be run from a user directory if needed. Applications from the Mac App Store are installed in the `/Applications` directory.

The `/Developer` directory is created when Xcode is installed. The `Developer` directory may be located in either the root (`/`), under `/Library` or embedded in the `Xcode.app` application in the `/Applications` directory. The `Developer` folder contains Apple Developer (Mac, iOS,) related data and resources.

Reference:

File System Programming Guide – File System Basics

https://developer.apple.com/library//mac/#/library/mac/documentation/FileManagement/Conceptual/FileSystemProgrammingGUide/FileSystemOverview/FileSystemOverview.html



The System Domain is used to store Apple specific system software.

The System Domain contains the system `Library` directory which contains files associated with Apple system resources.

You may have noticed a common theme in the directory structure. There are three `Library` directories on Mac OS X, each with its own purpose.

- User Library - `/Users/<username>/Library/`
- Local Library - `/Library/`
- System Library – `/System/Library/`

It is easy to get confused about which Library directory contains what data. This class will discuss the difference between each Library directory and the contents located within.

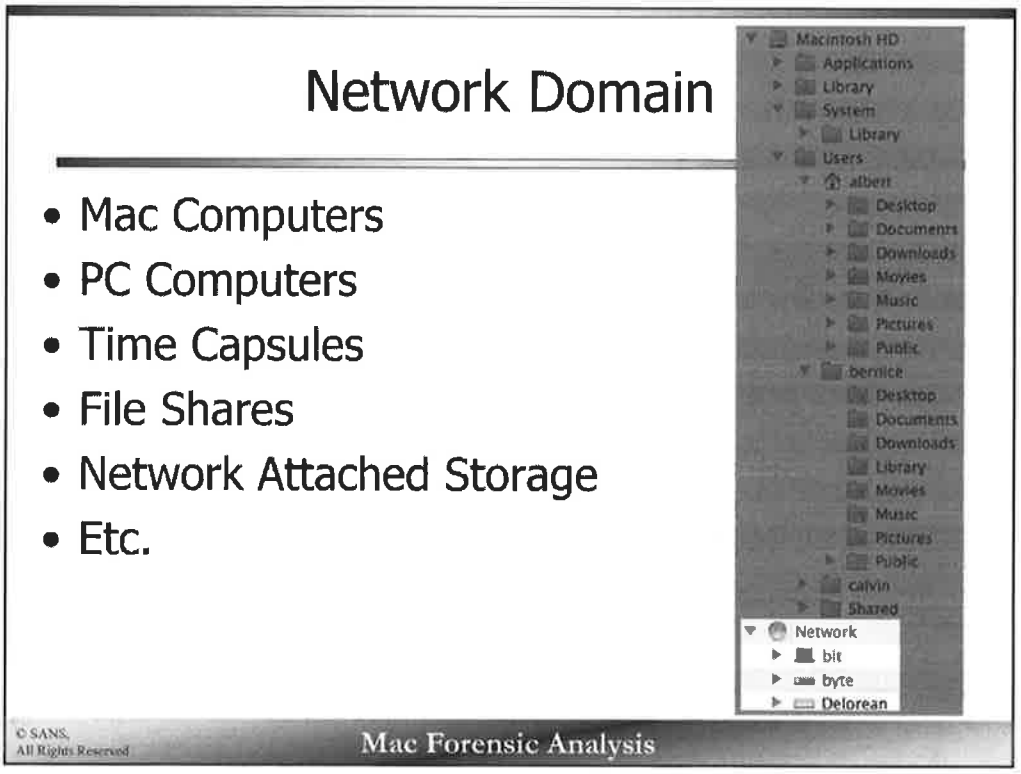
Reference:

File System Programming Guide – File System Basics

https://developer.apple.com/library//mac/#/library/mac/documentation/FileManagement/Conceptual/FileSystemProgrammingGUide/FileSystemOverview/FileSystemOverview.html

File System Program Guide – OS X Library Directory Details

https://developer.apple.com/library//mac/#/library/mac/documentation/FileManagement/Conceptual/FileSystemProgrammingGUide/MacOSXDirectories/MacOSXDirectories.html

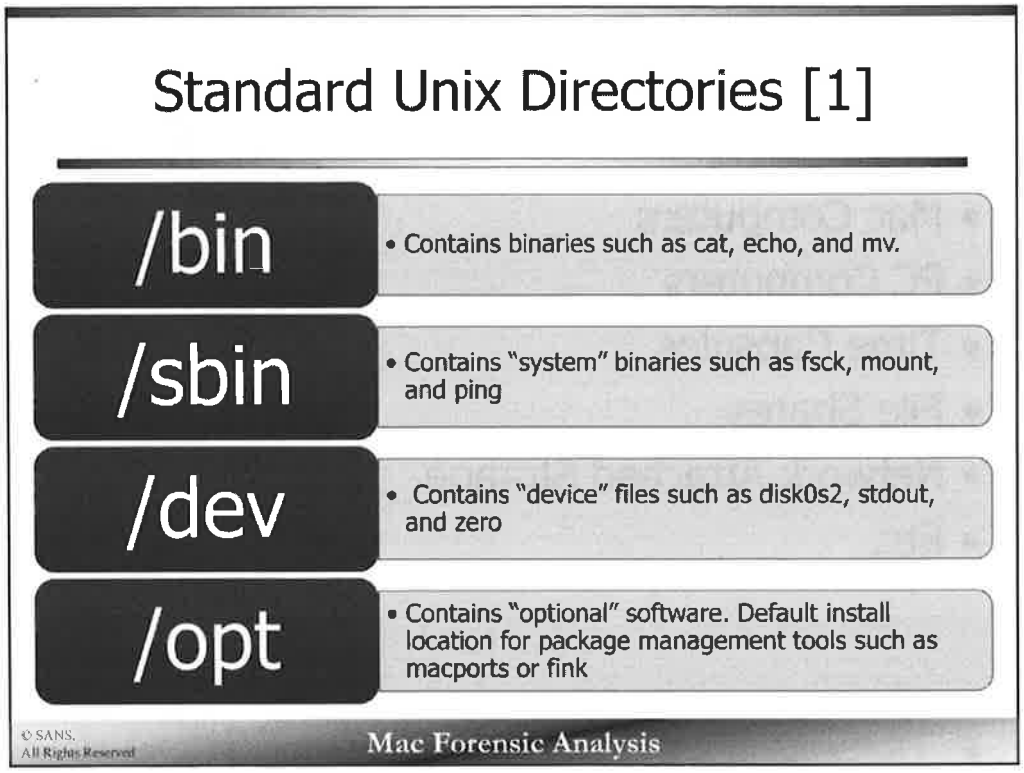


The Network Domain contains the network resources such as network area storage, Time Capsules, and other systems on the network.

Reference:

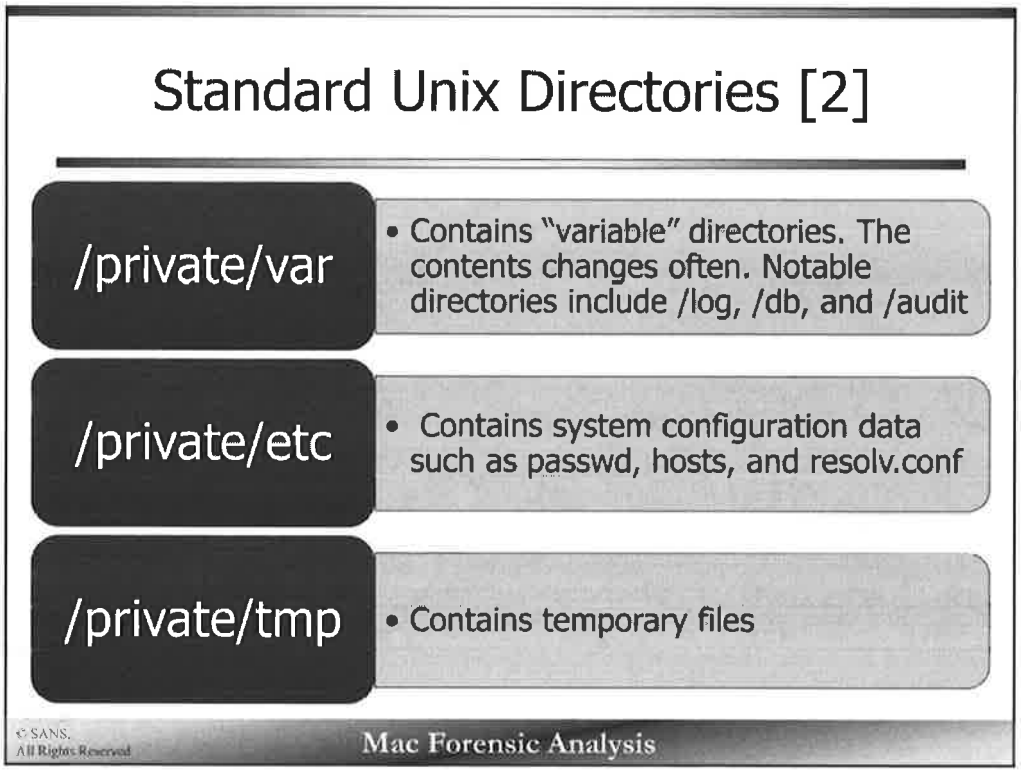
File System Programming Guide – File System Basics

https://developer.apple.com/library//mac/#/library/mac/documentation/FileManagement/Conceptual/FileSystemProgrammingGUide/FileSystemOverview/FileSystemOverview.html



Mac OS X systems contains standard Unix directories.

- `/bin` contains various command utilities
- `/sbin` contains contains system binaries
- `/dev` contains "device" files
- `/opt` contains "optional" software



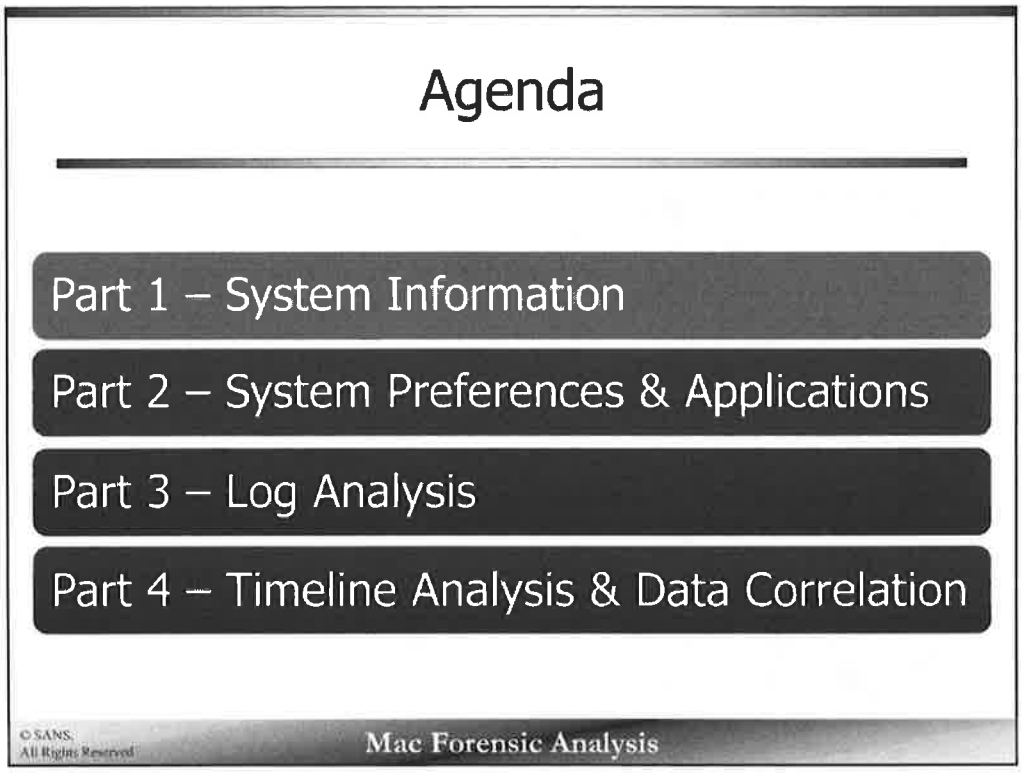
Mac OS X systems contains standard Unix directories.

`/private/var` contains notable "variable" directory such as `/log`, `/db`, and `/audit`.

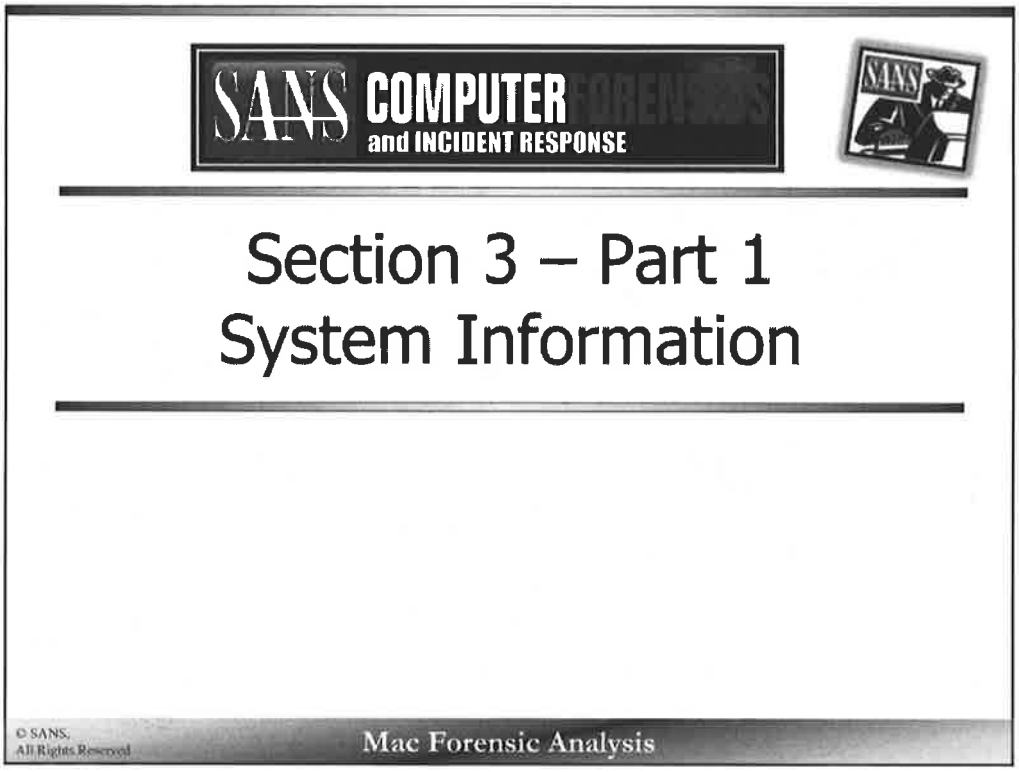
`/private/etc` contains system configuration data, with notable files such as `hosts`, `passwd`, and `resolv.conf`.

`/private/tmp` contains temporary files.

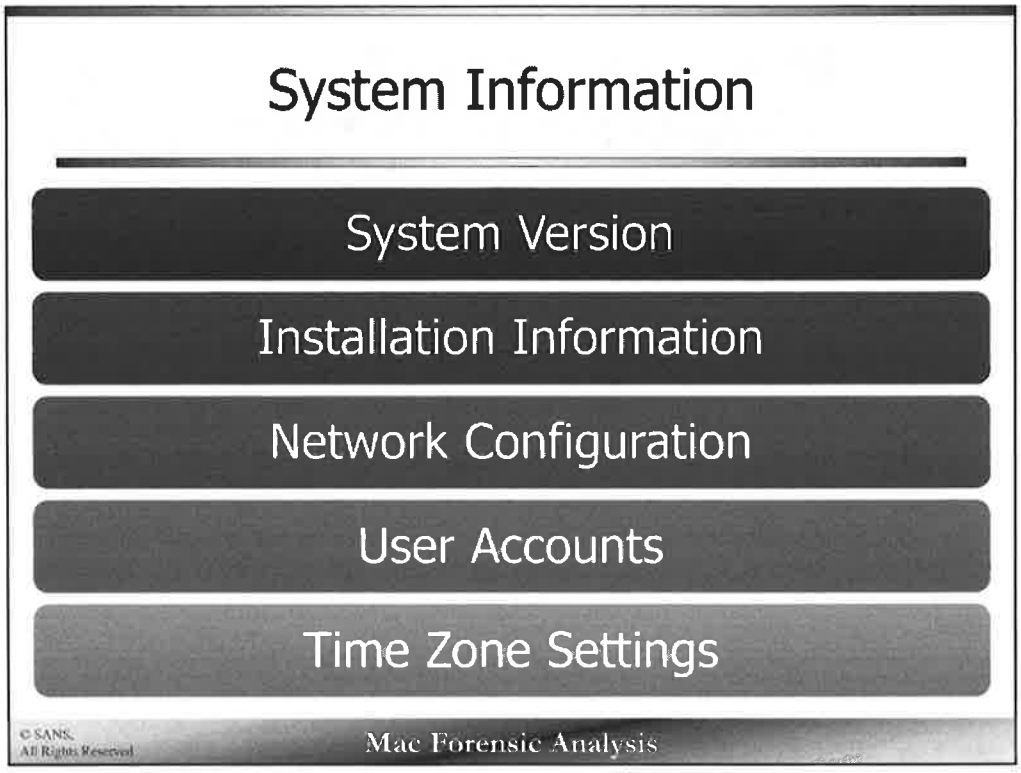
It is worth noting that the files `var`, `etc`, and `tmp` are all symbolic links to their `/private` equivalent.



This page intentionally left blank.

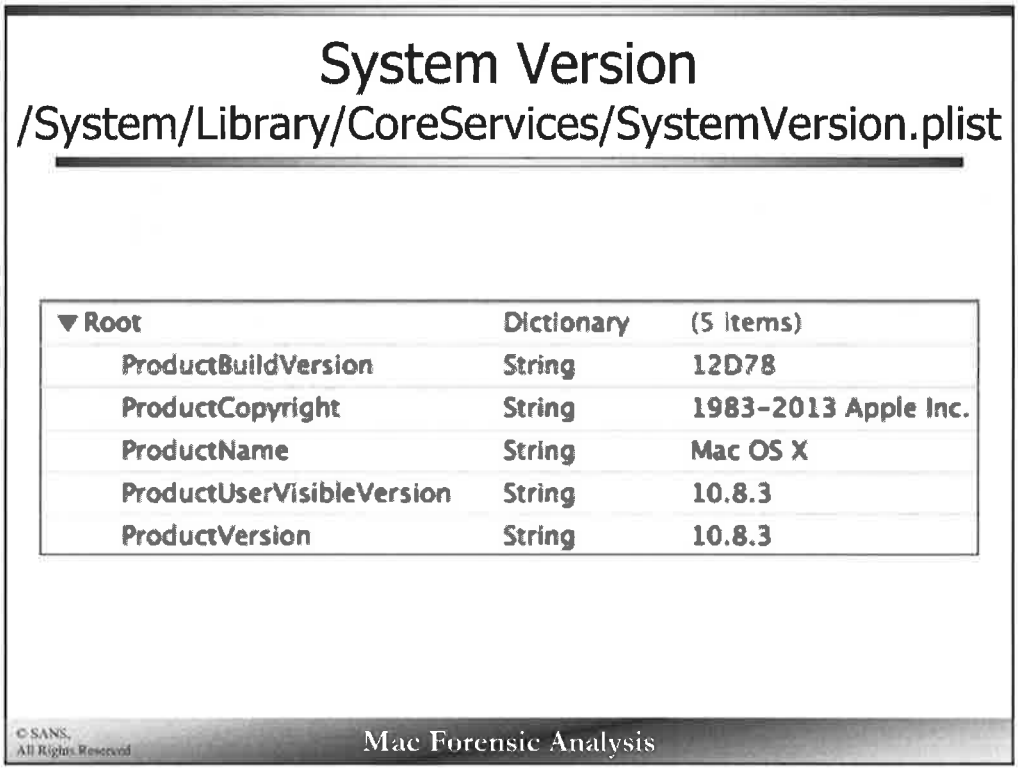


This page intentionally left blank.



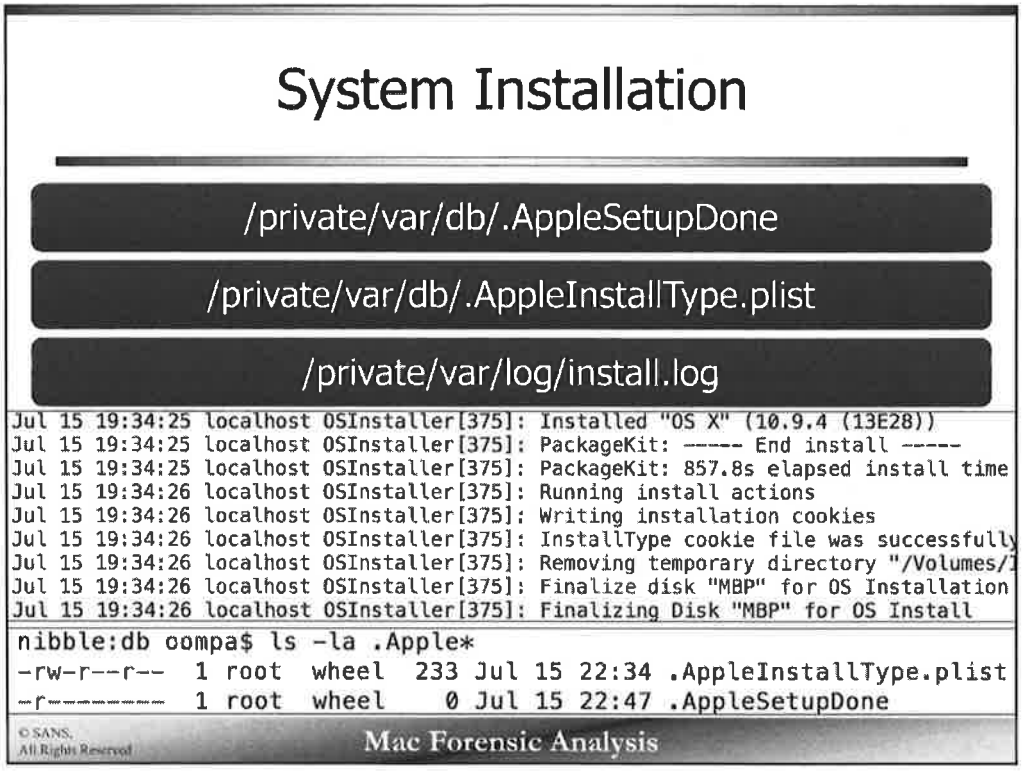
The `System` and `Local` domains contain initial triage information that can be useful to an investigator, such as:

- System Version
- Installation Information
- Network Configuration
- User Accounts
- Time Zone Settings



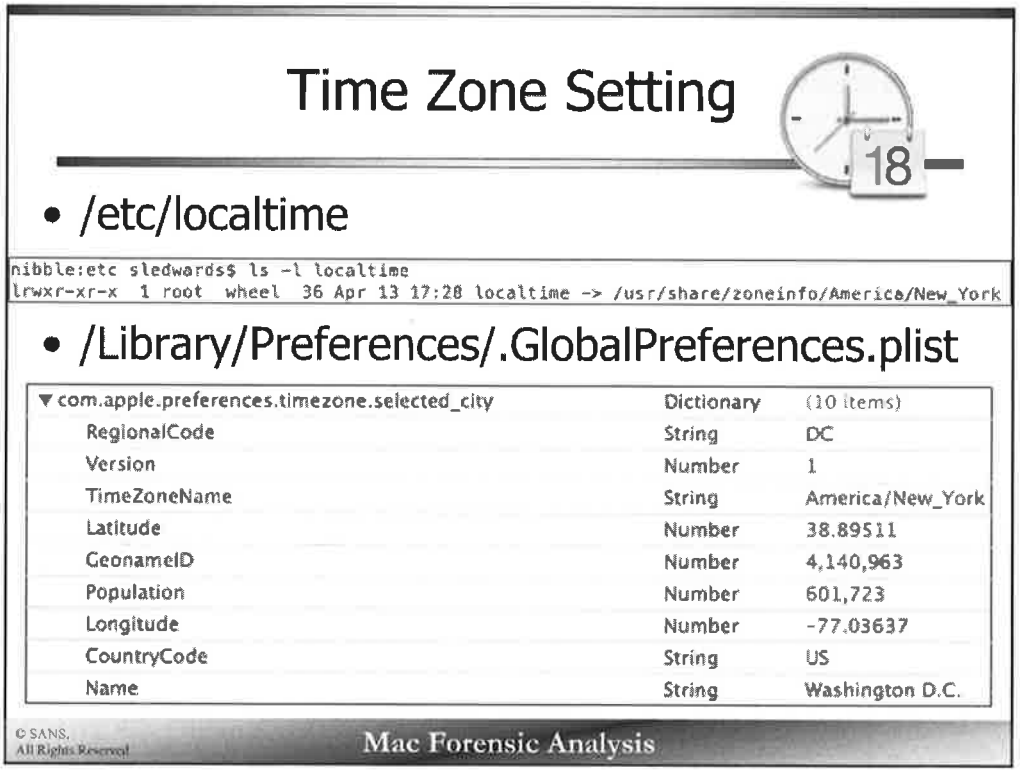
The `SystemVersion.plist` property list located in the `/System/Library/CoreServices/` directory contains the system version and build information.

In the example above, the system name and version is Mac OS X 10.8.3, while the build version is 12D78.



An original installation date may be found as the creation date for the `.AppleSetupDone` and `.AppleInstallType.plist` files located in the `/private/var/db/` directory.

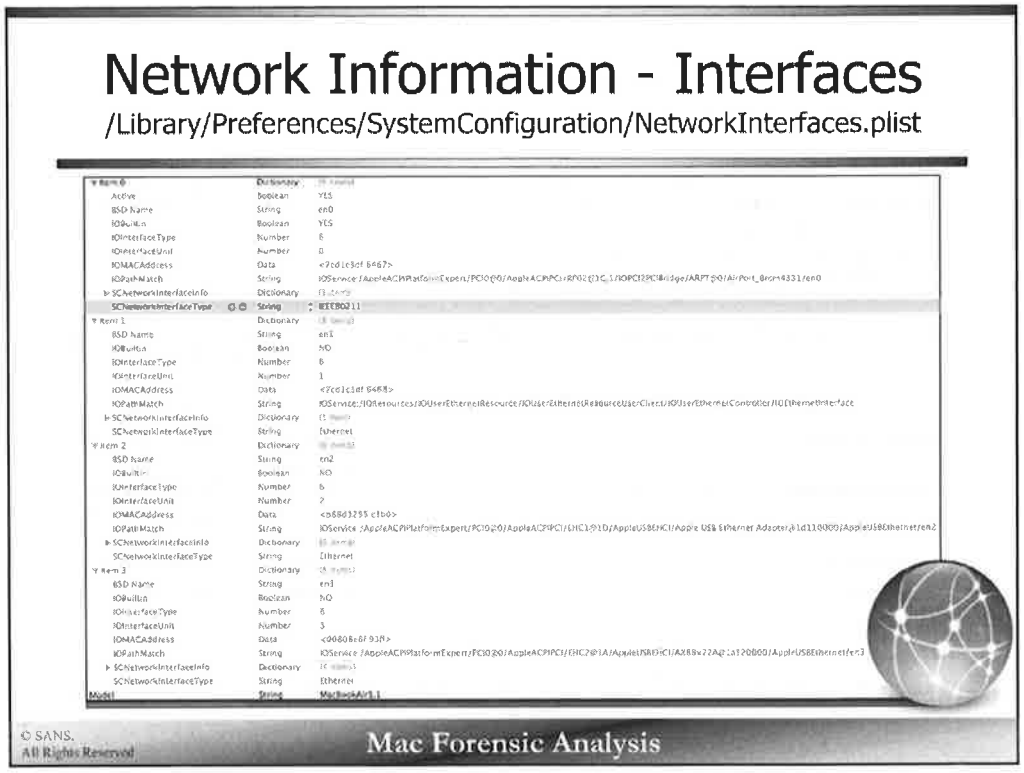
The OS X installation date may be able to be found in the `install.log` files located in the `/var/log` directory if the older files have not been turned over. These dates are when different versions of OS X were installed after the original installation.



The link to the file located in `/etc/localtime` contains the current time zone value for the system. In the screenshot you can see the local time zone is set for New York or the Eastern Standard Time zone.

The property list located in the `/Library/Preferences/` directory contains the time zone configuration data. The time zone is set for Washington, DC. This is for the same system configured for the New York time zone shown above. While the city may change, the time zone may be located in a more general location.

This configuration is likely chosen by the user when the system was configured during setup using the time zone map feature.



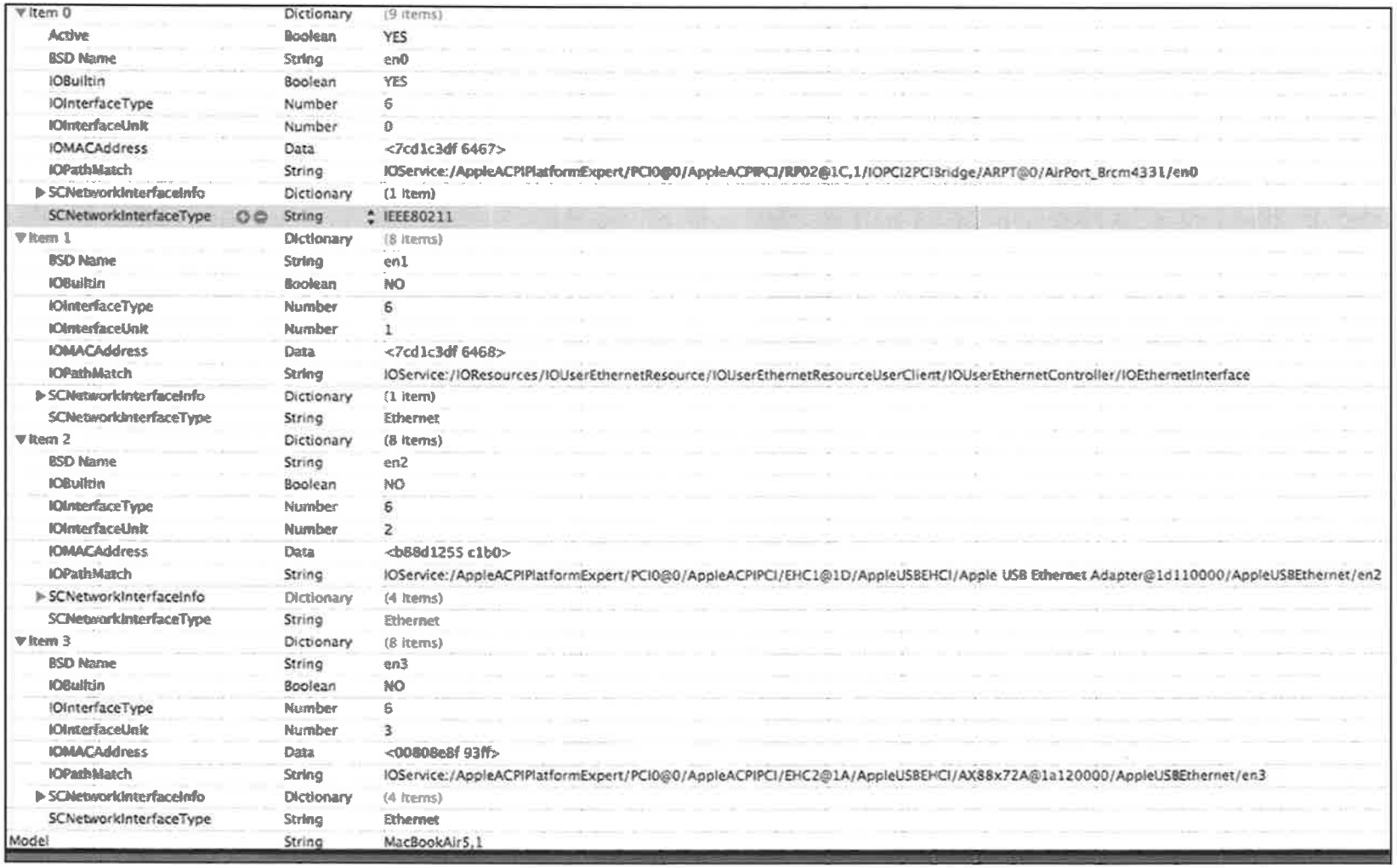
The `NetworkInterfaces.plist` file located in the `/Library/Preferences/SystemConfiguration/` directory contains the network interfaces for the system.

Each network interface on the system will have an `Item` key.

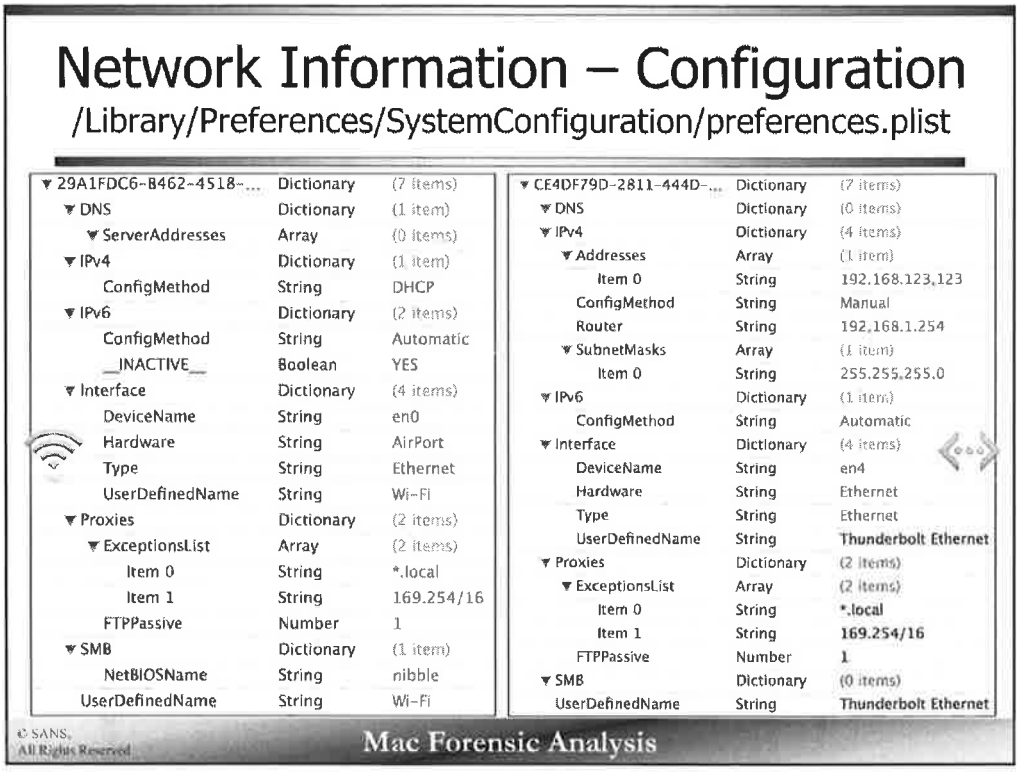
In the screenshot, the example has four network interfaces:

- `en0` – 802.11 Airport Card
- `en1` – Physical Ethernet interface -- This particular system does not have a physical Ethernet port as it is a MacBook Air. We can see this system is a MacBook Air from the "`Model`" key, shown at the bottom of the property list.
- `en2` – USB Ethernet Adapter – The MacBook Air can use a USB Ethernet adapter dongle as a wired Ethernet port.
- `en3` – USB Ethernet Adapter – A different Ethernet adapter used by this system.

You may also see a Thunderbolt Ethernet adapter in this list. Instead of using a USB port it will use the Thunderbolt port on the system.



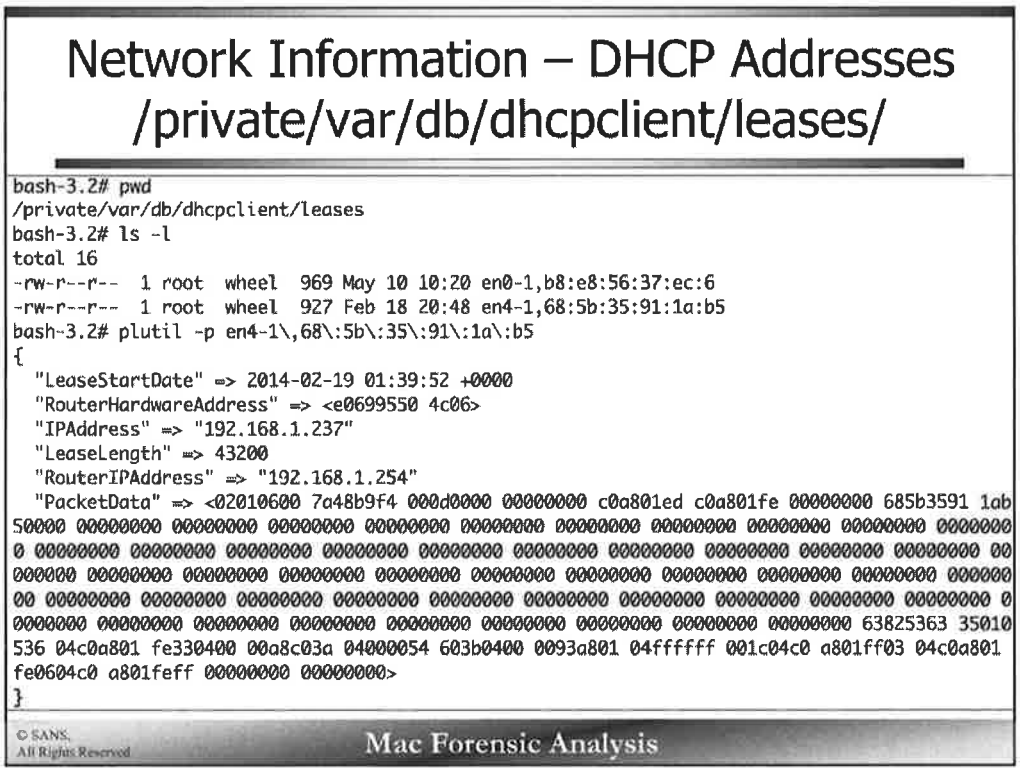
▼ Item 0	Dictionary	(9 items)
Active	Boolean	YES
BSD Name	String	en0
IOBuiltin	Boolean	YES
IOInterfaceType	Number	6
IOInterfaceUnit	Number	0
IOMACAddress	Data	<7cd1c3df 6467>
IOPathMatch	String	IOService:/AppleACPIPlatformExpert/PCI0@0/AppleACPIPCI/RP02@1C,1/IOPCI2PCIBridge/ARPT@0/AirPort_Brcm4331/en0
▶ SCNetworkInterfaceInfo	Dictionary	(1 item)
SCNetworkInterfaceType	String	IEEE80211
▼ Item 1	Dictionary	(8 items)
BSD Name	String	en1
IOBuiltin	Boolean	NO
IOInterfaceType	Number	6
IOInterfaceUnit	Number	1
IOMACAddress	Data	<7cd1c3df 6468>
IOPathMatch	String	IOService:/IOResources/IOUserEthernetResource/IOUserEthernetResourceUserClient/IOUserEthernetController/IOEthernetInterface
▶ SCNetworkInterfaceInfo	Dictionary	(1 item)
SCNetworkInterfaceType	String	Ethernet
▼ Item 2	Dictionary	(8 items)
BSD Name	String	en2
IOBuiltin	Boolean	NO
IOInterfaceType	Number	6
IOInterfaceUnit	Number	2
IOMACAddress	Data	<b88d1255 c1b0>
IOPathMatch	String	IOService:/AppleACPIPlatformExpert/PCI0@0/AppleACPIPCI/EHC1@1D/AppleUSBEHCI/Apple USB Ethernet Adapter@1d110000/AppleUSBEthernet/en2
▶ SCNetworkInterfaceInfo	Dictionary	(4 items)
SCNetworkInterfaceType	String	Ethernet
▼ Item 3	Dictionary	(8 items)
BSD Name	String	en3
IOBuiltin	Boolean	NO
IOInterfaceType	Number	6
IOInterfaceUnit	Number	3
IOMACAddress	Data	<00808e8f 93ff>
IOPathMatch	String	IOService:/AppleACPIPlatformExpert/PCI0@0/AppleACPIPCI/EHC2@1A/AppleUSBEHCI/AX88x72A@1a120000/AppleUSBEthernet/en3
▶ SCNetworkInterfaceInfo	Dictionary	(4 items)
SCNetworkInterfaceType	String	Ethernet
Model	String	MacBookAir5,1



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

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

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

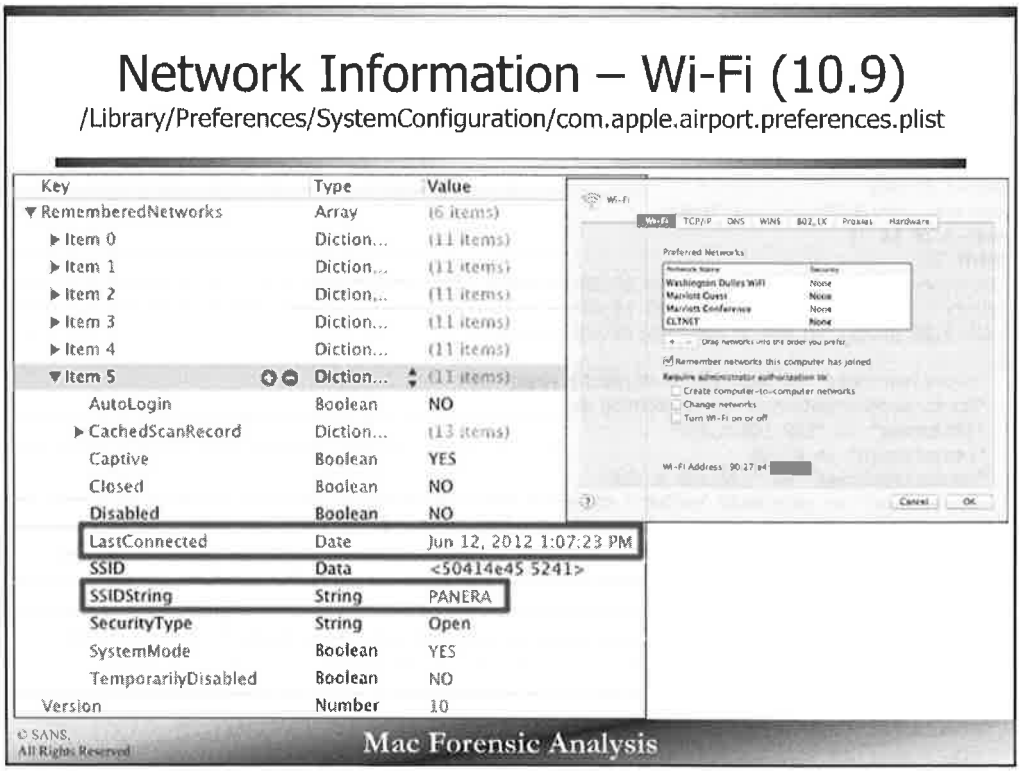


The XML-based files located in the `/private/var/db/dhcpclient/leases/` directory contains files with configurations for DHCP network settings. These contain the settings for the latest connection on the specified interface.

Shown in the screenshot above, the example shows two DHCP configurations. One for the en0 adapter and another for the en4 adapter. Each adapter has an associated MAC address in the filename.

Each file contains:

- Lease Start Date
- Router MAC Address
- Assigned IP Address
- SSID of Access Point (Wi-Fi only)
- DHCP Lease Length (in minutes)
- Router IP Address
- Packet Data



The `com.apple.airport.preferences.plist` property list file located in the `/Library/Preferences/SystemConfiguration/` directory contains network information of the "remembered" or "saved" networks. Each network is stored in its own `Item` key.

The "remembered" networks are a list of networks the system has previously established a connection; These items do not appear to be purged unless performed by the user. If removed using the GUI shown, the items will be removed from the property list.

Some of the attributes available for each network include:

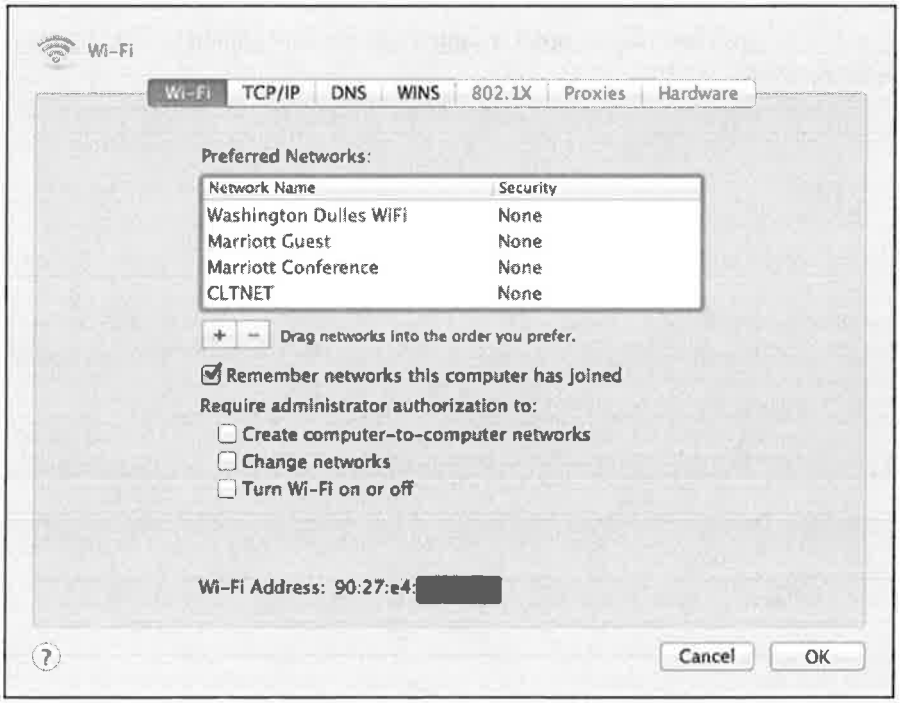
- Captive (that popup screen you get in hotels and restaurants)
- When the system last connected to the network, stored in local system time.
- Network SSID
- Automatic logon

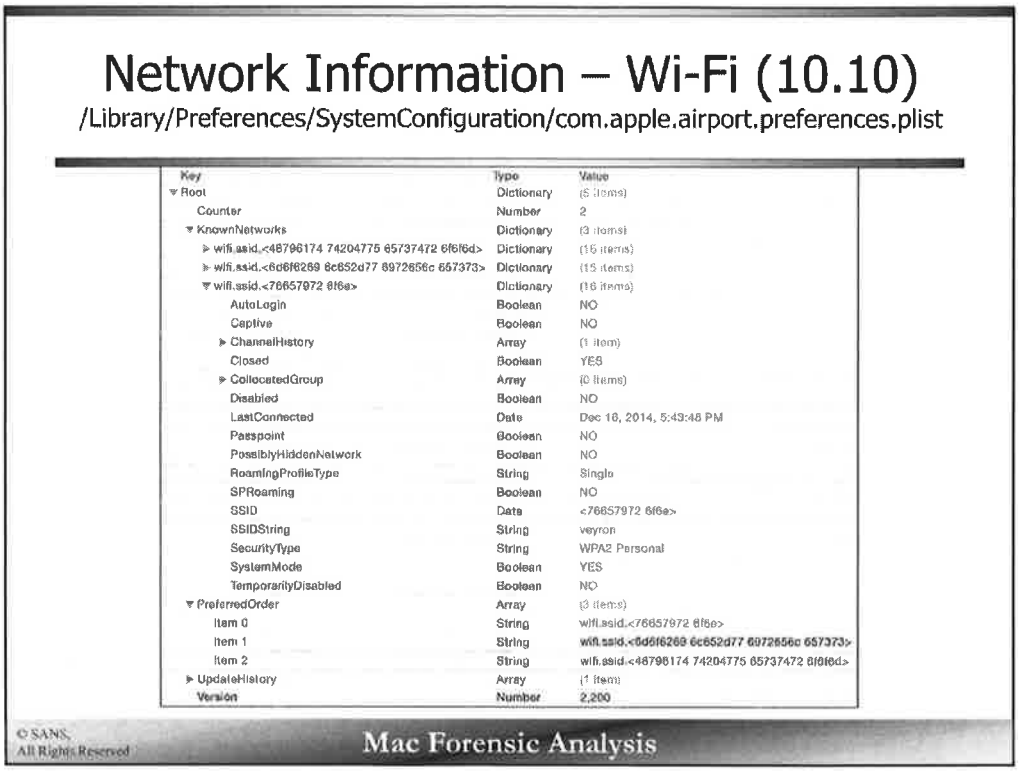
These networks can help an investigator determine where a system might have travelled to if the network BSSIDs are unique such as a restaurant or hotel name.

Note: This file will only be available if the system has a wireless network card available.

By default, these are stored in a chronological format (Item 0 is the oldest), however the user can use the GUI to change the order in which they attempt to connect which will change the order in the property list file.

Key	Type	Value
▼ RememberedNetworks	Array	(6 items)
▶ Item 0	Diction...	(11 items)
▶ Item 1	Diction...	(11 items)
▶ Item 2	Diction...	(11 items)
▶ Item 3	Diction...	(11 items)
▶ Item 4	Diction...	(11 items)
▼ Item 5	Diction...	(11 items)
AutoLogin	Boolean	NO
▶ CachedScanRecord	Diction...	(13 items)
Captive	Boolean	YES
Closed	Boolean	NO
Disabled	Boolean	NO
LastConnected	Date	Jun 12, 2012 1:07:23 PM
SSID	Data	<50414e45 5241>
SSIDString	String	PANERA
SecurityType	String	Open
SystemMode	Boolean	YES
TemporarilyDisabled	Boolean	NO
Version	Number	10



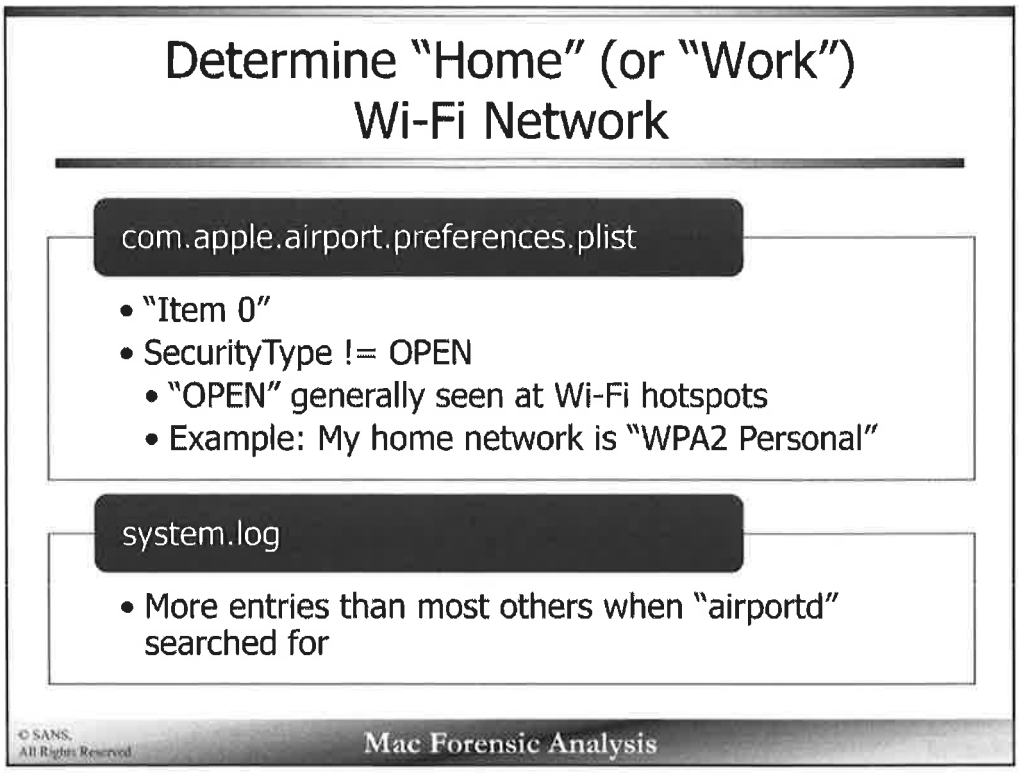


On 10.10 systems the `com.apple.airport.preferences.plist` property list file contains similar information, however is organized a bit differently.

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

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

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

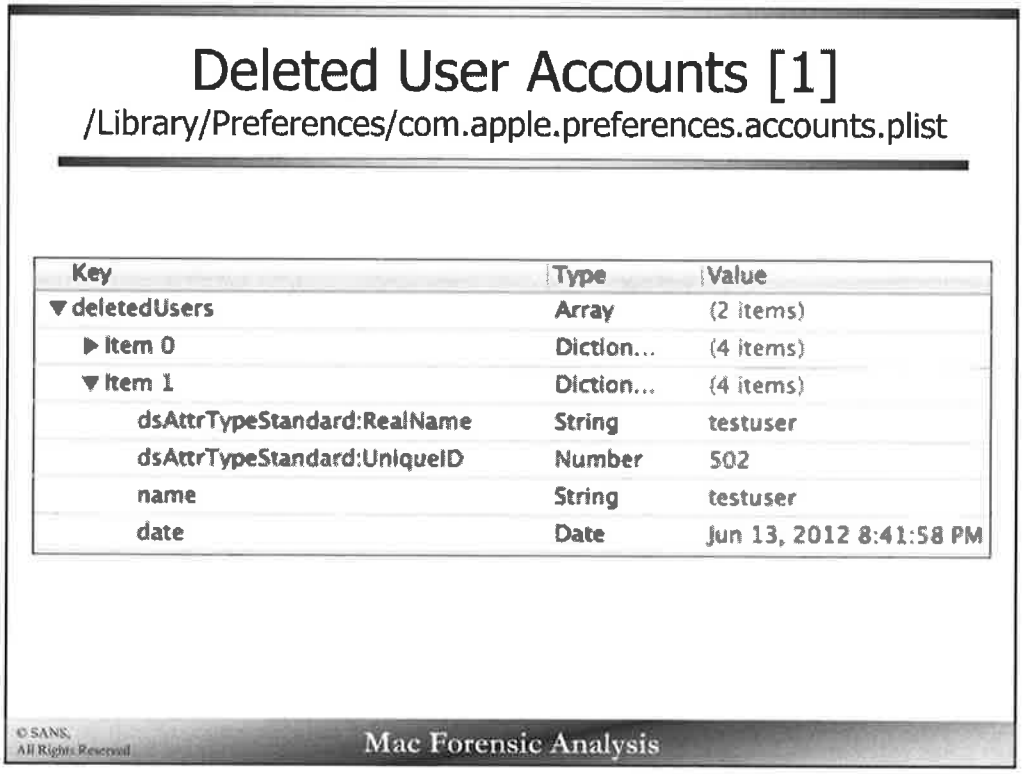


To determine which one of the listed networks is the "home" network, such as the system owner's home access point or the company enterprise Wi-Fi, analysis can be done on the airport property list file `com.apple.airport.preferences.plist` or the `system.log`. This is assuming the user is using Wi-Fi versus a wired connection.

The airport preferences file should at least have one `Item` key in the list if the wireless network adapter was used. When a system is setup, the first item or `Item 0` is populated. Most users set up their new laptop or device in their own homes and offices. Security of these access points should (hopefully) be secured and not listed as "`OPEN`" as most Wi-Fi hotspots such as the local coffee shop.

The `system.log` file can be searched for the "`airportd`" term to determine where a device has established the most connections. Many users tend to use the device in one location more than any other such as their home or office.

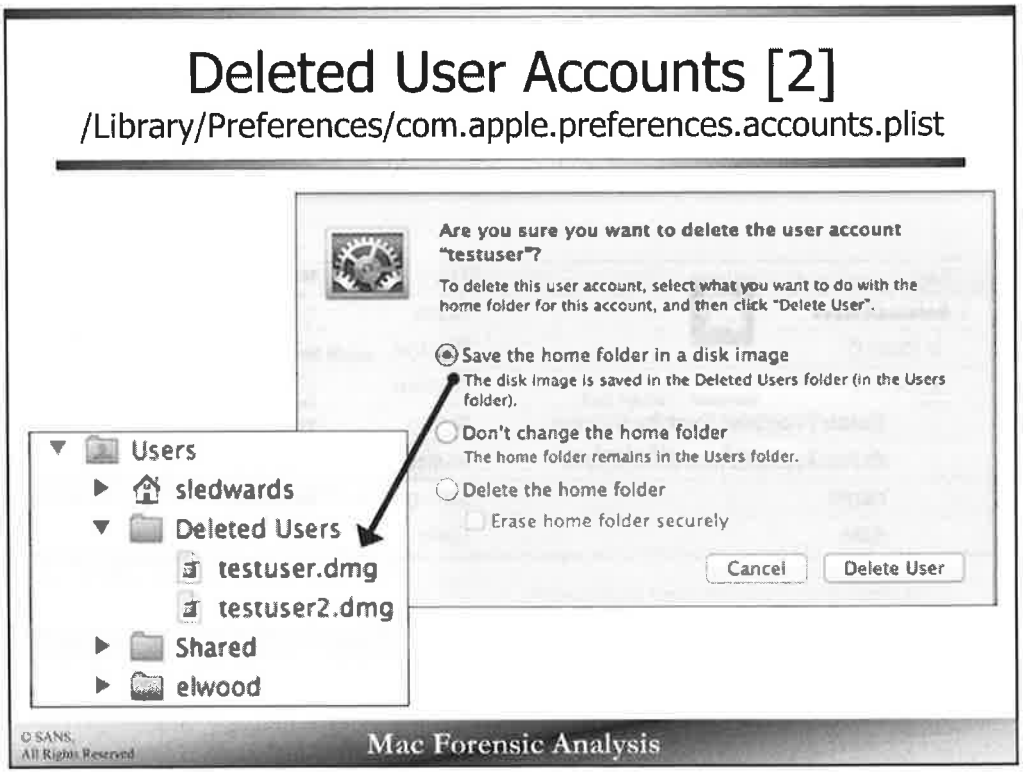
This is not exactly a scientific method to determine a home network, but can help in determining the user profile of a system user.



The deleted users are shown in the `com.apple.preferences.accounts.plist` file under the `deletedUsers` key. This property list is located in the `/Library/Preferences/` directory.

This key contains:

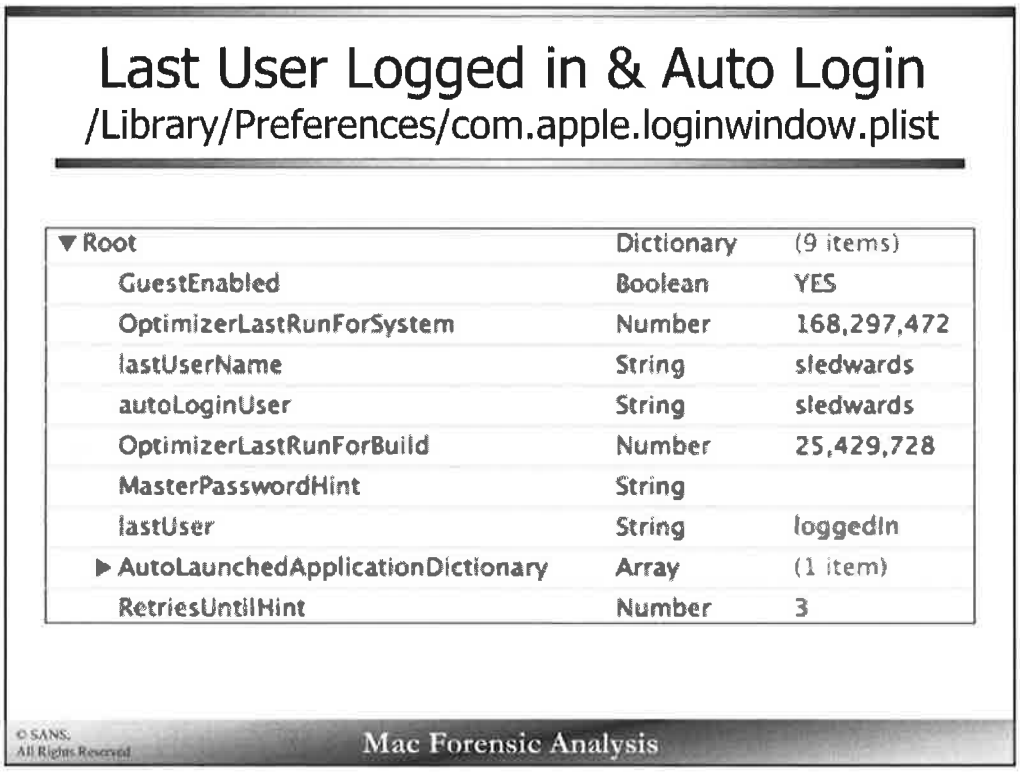
- Deleted user's "Real Name"
- UID
- Username
- Deletion date (local system time)



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

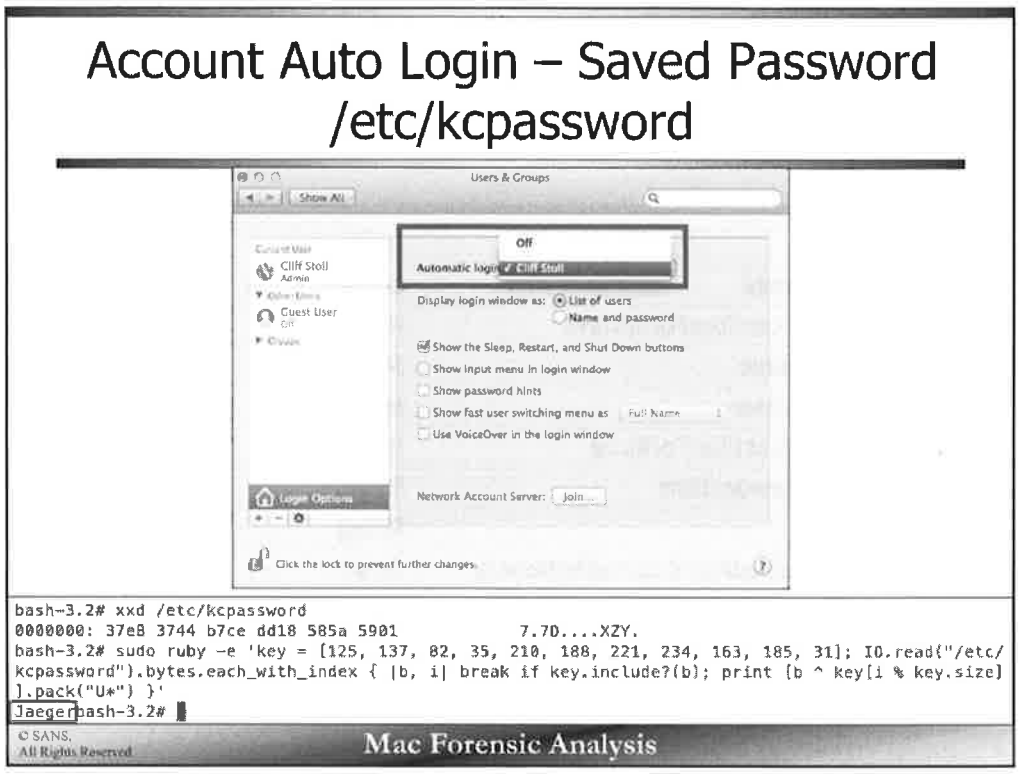
- "`Save the home folder in a disk image`" – This default option archives the user's home directory and saves it in an disk image file (DMG). The inset screenshot shows a couple of deleted user accounts saved to DMG files which are then moved to the `/Users/Deleted Users/` directory.
- "`Don't change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

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



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

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



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

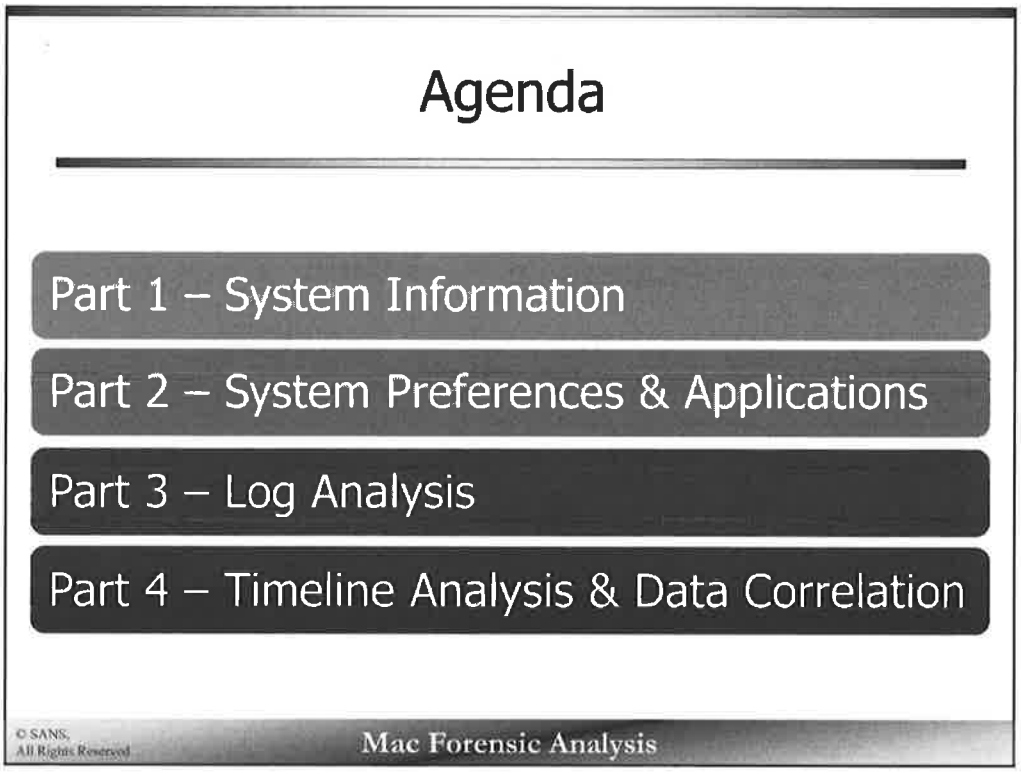
The user's password is then 0R'd with a multi-byte key and stored in `/etc/kcpassword`. To decode this password use the Ruby script below, provided by Lauri Ranta on apple.stackexchange.com.

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

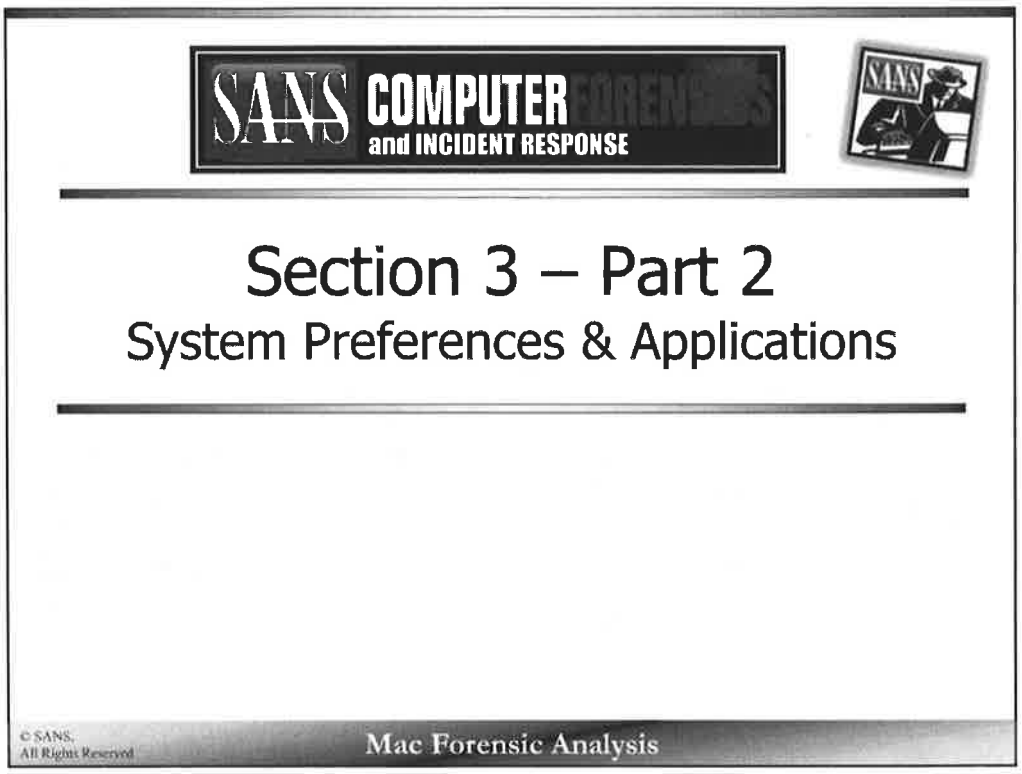
References:

http://www.brock-family.org/gavin/perl/kcpassword.html

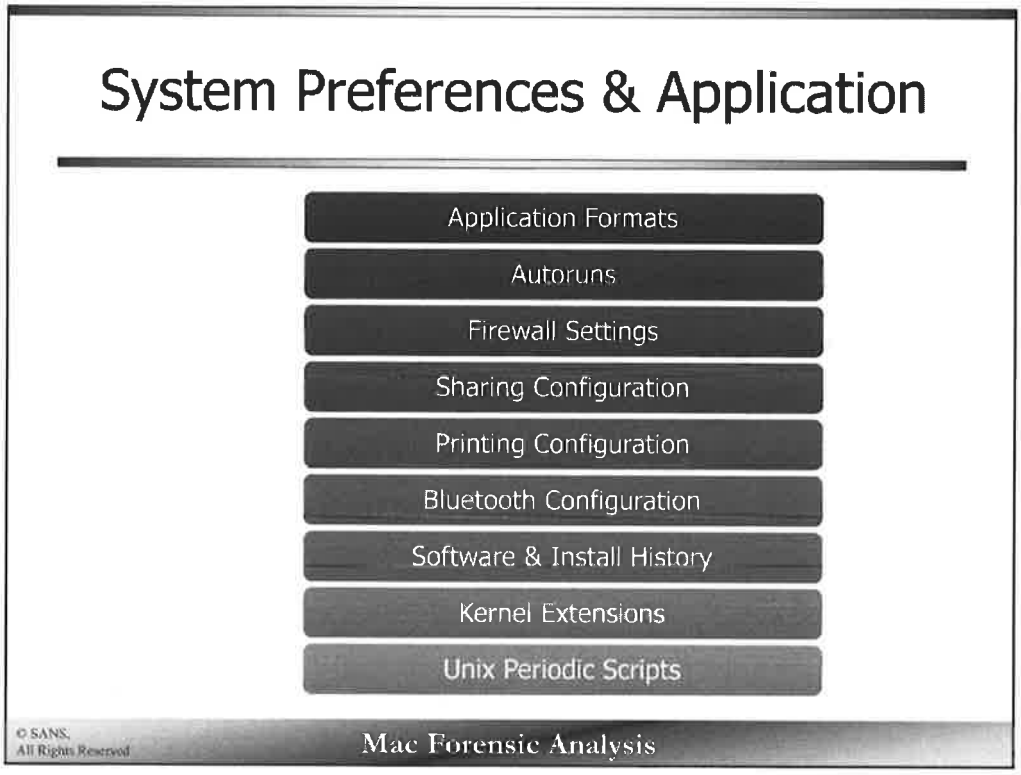
http://apple.stackexchange.com/questions/50652/does-activating-auto-login-compromise-secure-password-storage



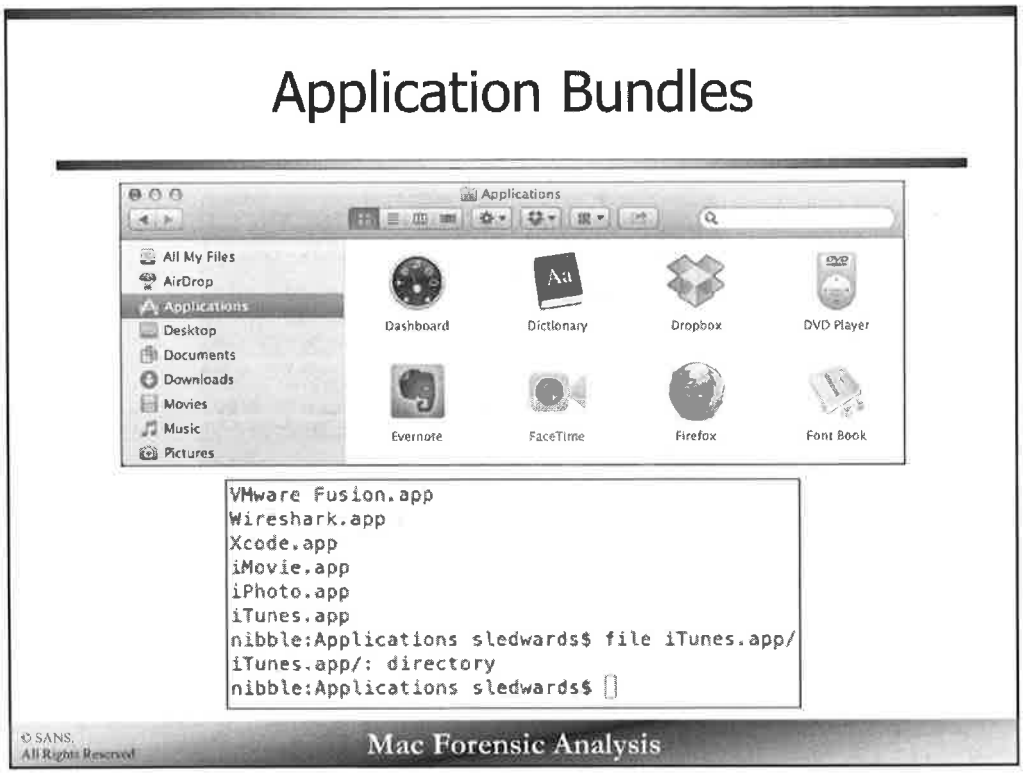
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.



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 is 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.

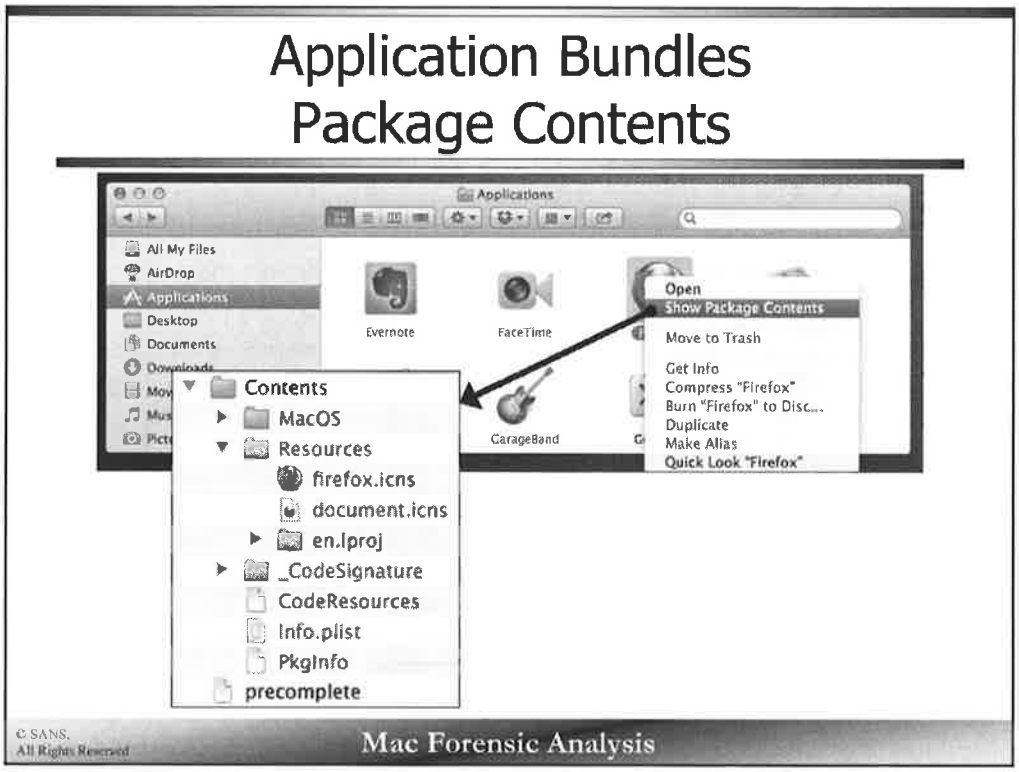
References:

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 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

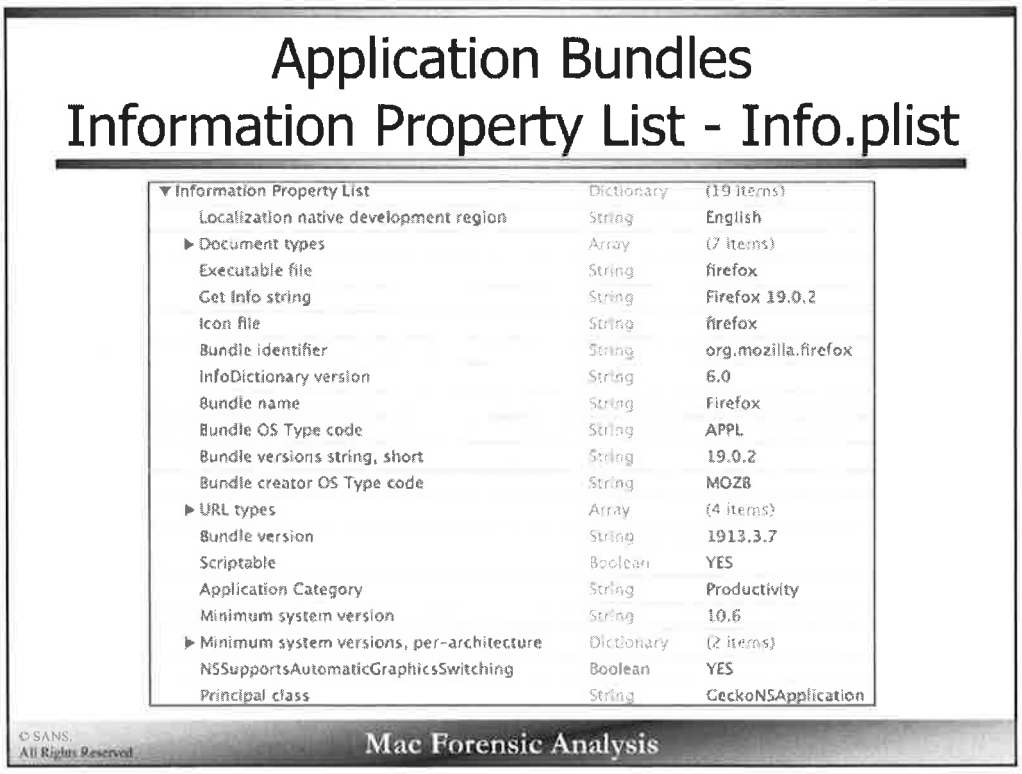
References:

Bundle Programming Guide – Apple Developer Documentation

http://developer.apple.com/library/mac/#documentation/CoreFoundation/Conceptual/CFBundles/AboutBundles/AboutBundles.html

Application Bundles – Apple Developer Documentation

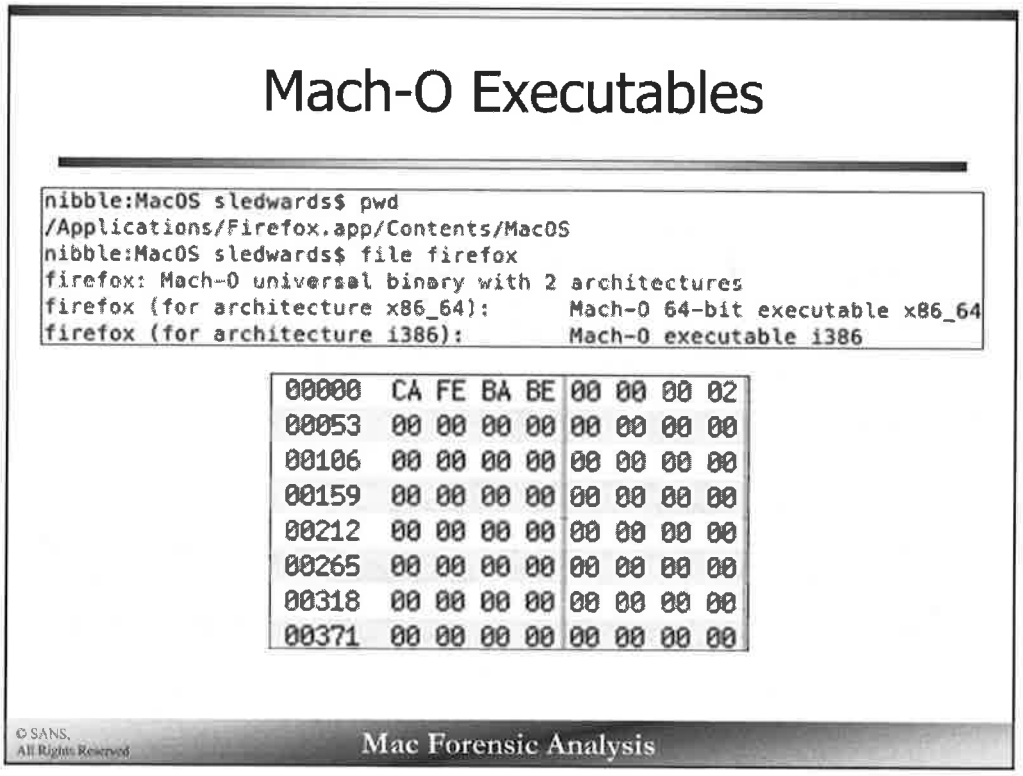
http://developer.apple.com/library/mac/#documentation/CoreFoundation/Conceptual/CFBundles/BundleTypes/BundleTypes.html#//apple_ref/doc/uid/10000123i-CH101-SW13



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 architectures types.



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 with 32 and 64-bit Intel processors.

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:

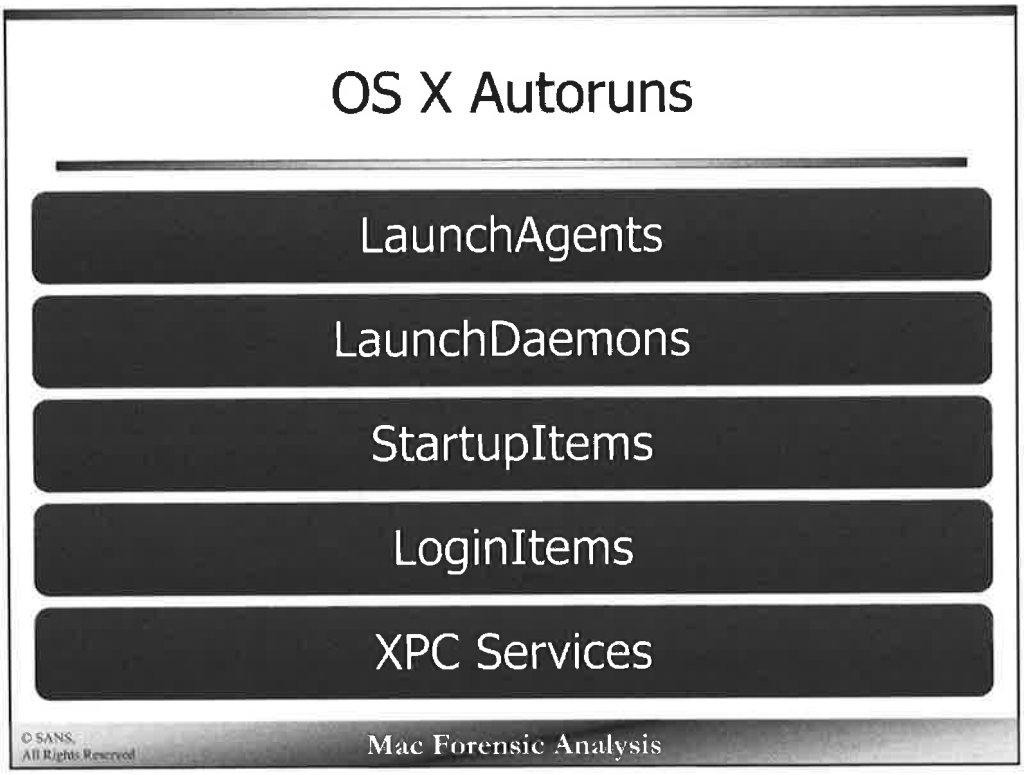
- `0xCAFEBABE` – Fat binary
- `0xFEEDFACE` – 32-bit
- `0xFEEDFACF` – 64-bit
- `0xCEFAEDFE` – 32-bit, little-endian
- `0xCFFAEDFE` – 64-bit, little-endian

References:

OS X Mach-O File Format Reference – Apple Developer Documentation

https://developer.apple.com/library/mac/#documentation/DeveloperTools/Conceptual/MachORuntime/Reference/reference.html

http://appleinsider.com/articles/08/10/28/road_to_mac_os_x_snow_leopard_64_bit_to_the_kernel.html



Mac OS X 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. A python script called OS X Autoruns is available from Malicious Streams [http://www.malicious-streams.com/Downloads/files/6bad73d5437c84b23fde9ebad97c7cff-6.html] that enumerates many of these locations. It is worth noting it does not enumerate the XPC Services.

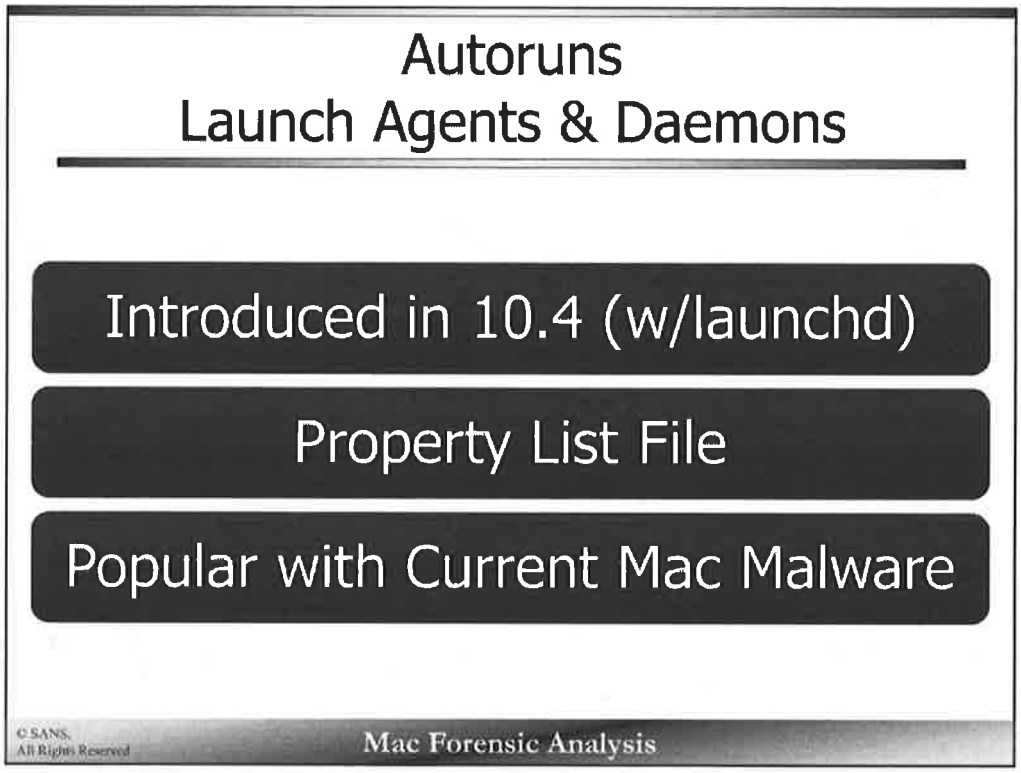
Some tools you might want to look at to find autoruns and other auditing information:

- Knockknock - github.com/synack/knockknock
- OSXAuditor - github.com/jipegit/OSXAuditor
- checkout4mac - code.google.com/p/checkout4mac/
- pac4mac - code.google.com/p/pac4mac/

References:

Mac OS X Malware Analysis - Joel Yonts, Malicious Streams

http://www.malicious-streams.com//article/Mac_OSX_Malware_Analysis.pdf



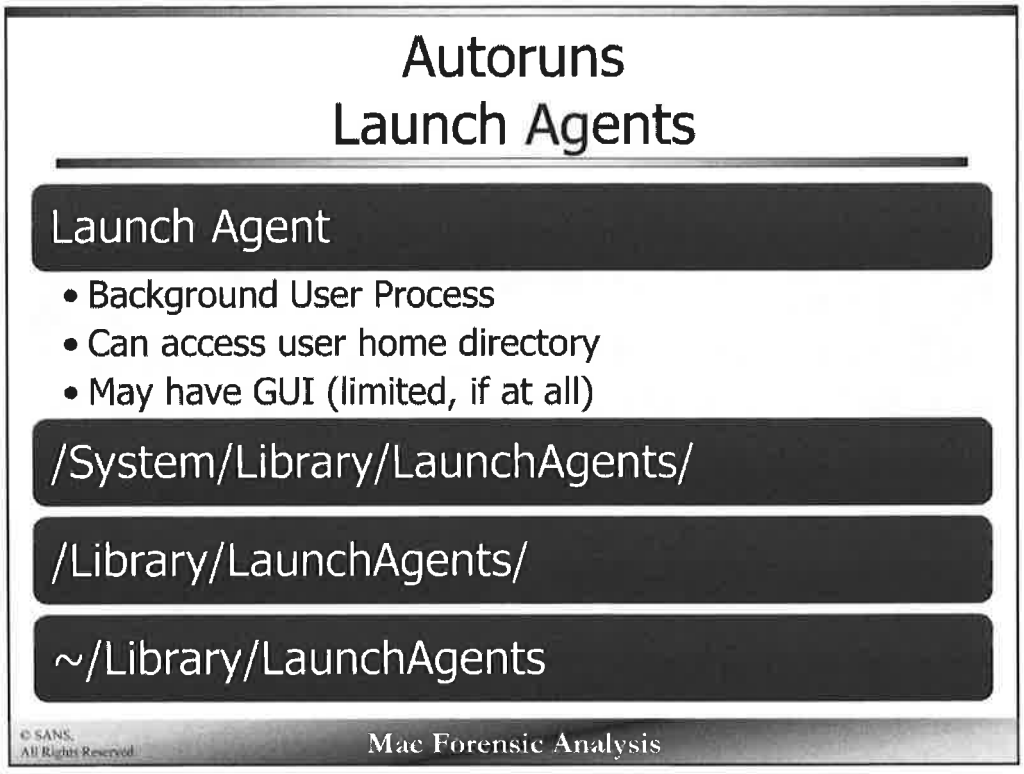
The preferred method of creating persistence is by using Launch Agents and Daemons. These methods were introduced in Mac OS X 10.4 with the introduction of `launchd`, the system agent and daemon launch manager.

The agents and daemons are implemented as information in a property list file. Most modern Mac malware uses these methods to keep their malicious files persistent across system shutdown and reboots.

Reference:

Daemons and Agents - TechNote 2083 (TN2083)

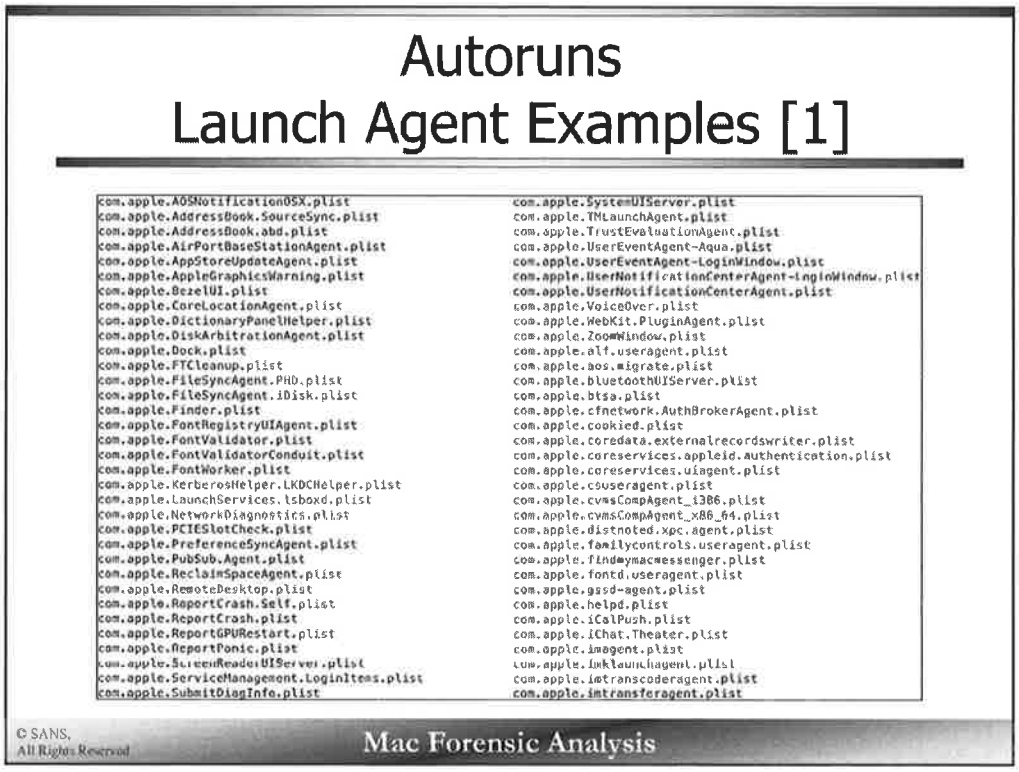
http://developer.apple.com/library/mac/#technotes/tn2083/_index.html



Launch Agents are a background user process while daemons are background system processes.

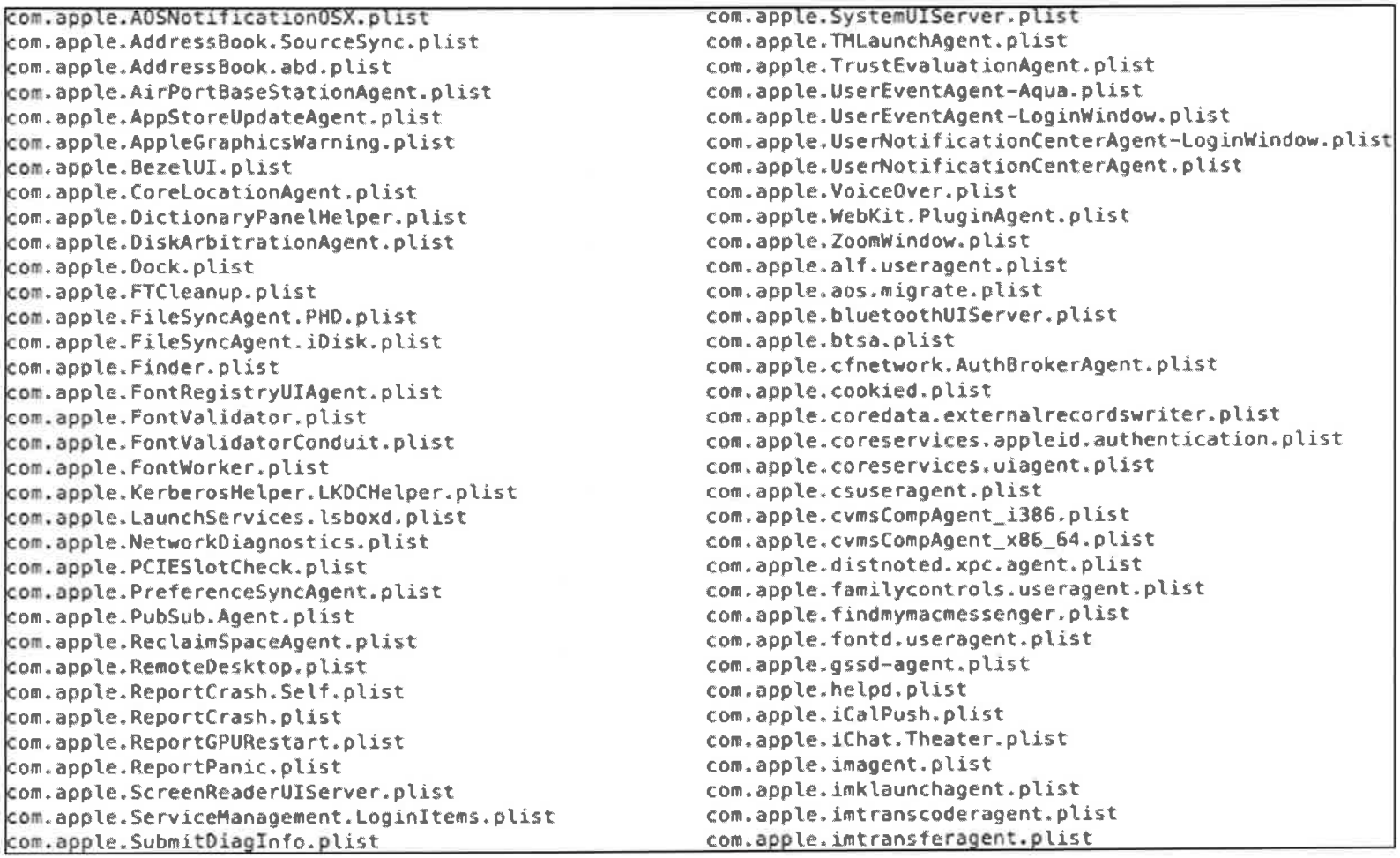
These processes can access user home directories and have limited GUI interfaces while daemons cannot. The typical locations where `LaunchAgents` are found is the `LaunchAgents` directory located in the System Library, Local Library, and User Library directories.

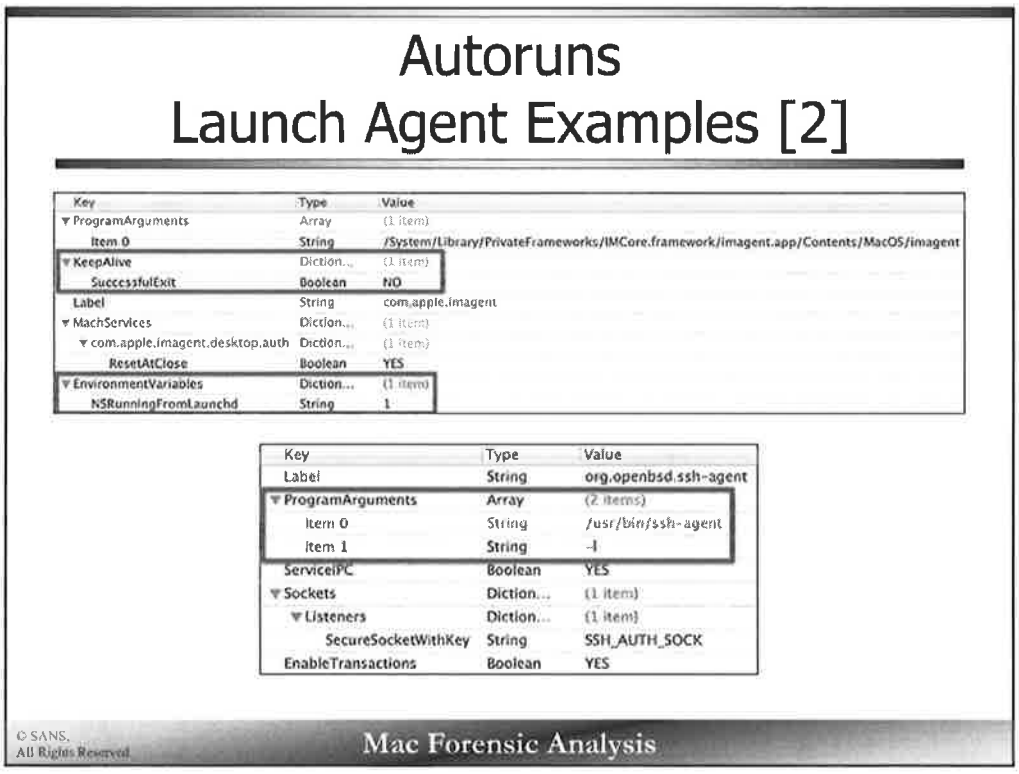
Those agents launched from `the /System/Library` or `/Library/` directories are launched for all users, while those located in the Users Library directory are available only to that user.



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



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

- `Program` or `ProgramArguments` – File path of the program to launch
- `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 top example shows a launch agent for the "`imagent`" process. The `KeepAlive` key contains the subkey "`SucessfulExit`". According to TechNote 2083 this means to "run on demand as long as you exit successfully." This example also shows the `EnvironmentVariables` which is used to specify environment variables.

The middle screenshot shows a good example of the `ProgramArguments` key. 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.

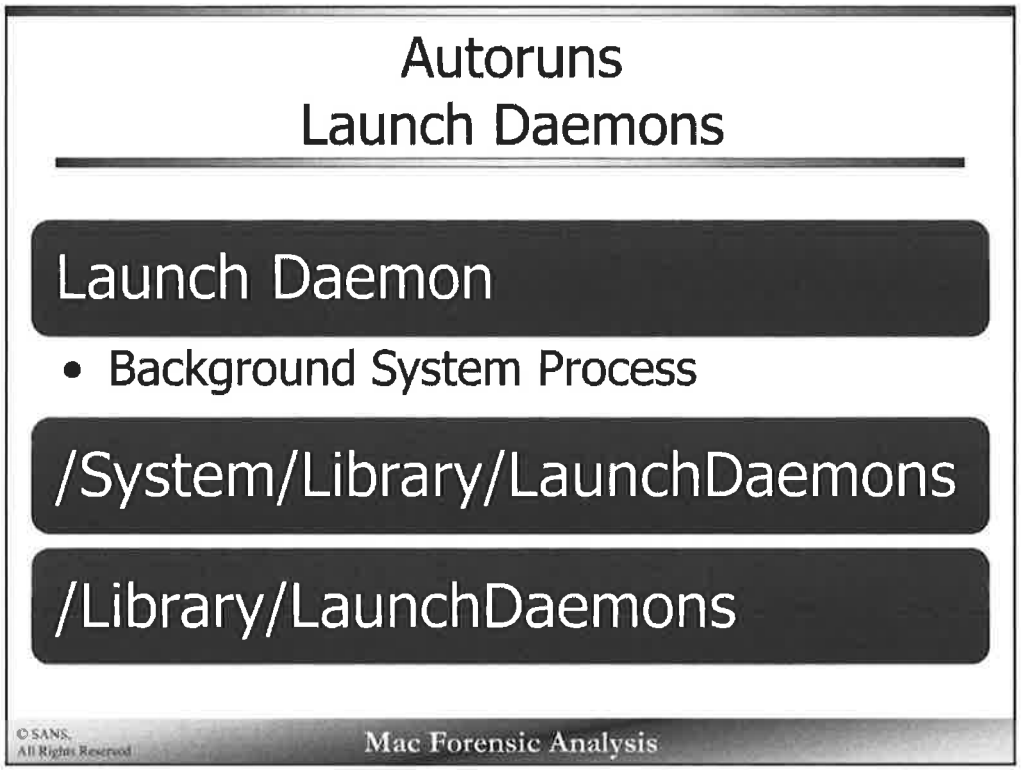
References:

launchd.plist Man Page:

https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man5/launchd.plist.5.html

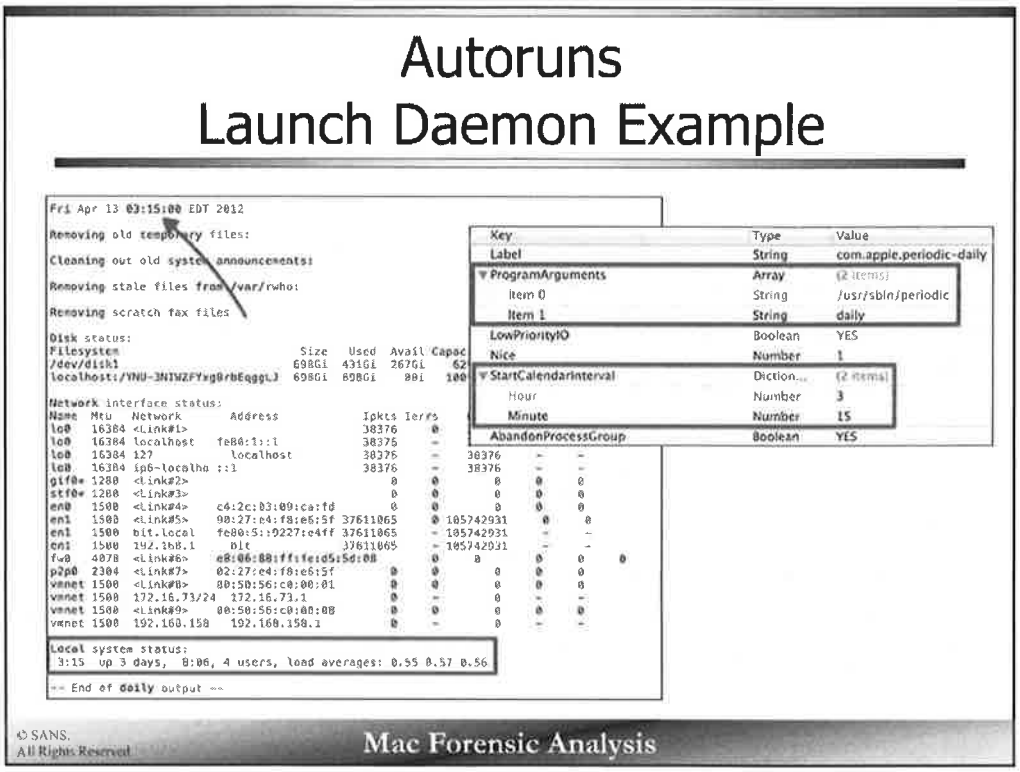
Key	Type	Value
▼ ProgramArguments	Array	(1 item)
Item 0	String	/System/Library/PrivateFrameworks/IMCore.framework/imagent.app/Contents/MacOS/imagent
▼ KeepAlive	Diction...	(1 item)
SuccessfulExit	Boolean	NO
Label	String	com.apple.imagent
▼ MachServices	Diction...	(1 item)
▼ com.apple.imagent.desktop.auth	Diction...	(1 item)
ResetAtClose	Boolean	YES
▼ EnvironmentVariables	Diction...	(1 item)
NSRunningFromLaunchd	String	1

Key	Type	Value
Label	String	org.openbsd.ssh-agent
▼ ProgramArguments	Array	(2 items)
Item 0	String	/usr/bin/ssh-agent
Item 1	String	-l
ServiceIPC	Boolean	YES
▼ Sockets	Diction...	(1 item)
▼ Listeners	Diction...	(1 item)
SecureSocketWithKey	String	SSH_AUTH_SOCK
EnableTransactions	Boolean	YES



Launch Daemons are a background system process while agents are background user processes.

The typical locations where Launch Daemons are found is the `LaunchDaemons` directory located in the System Library and Local Library directories.



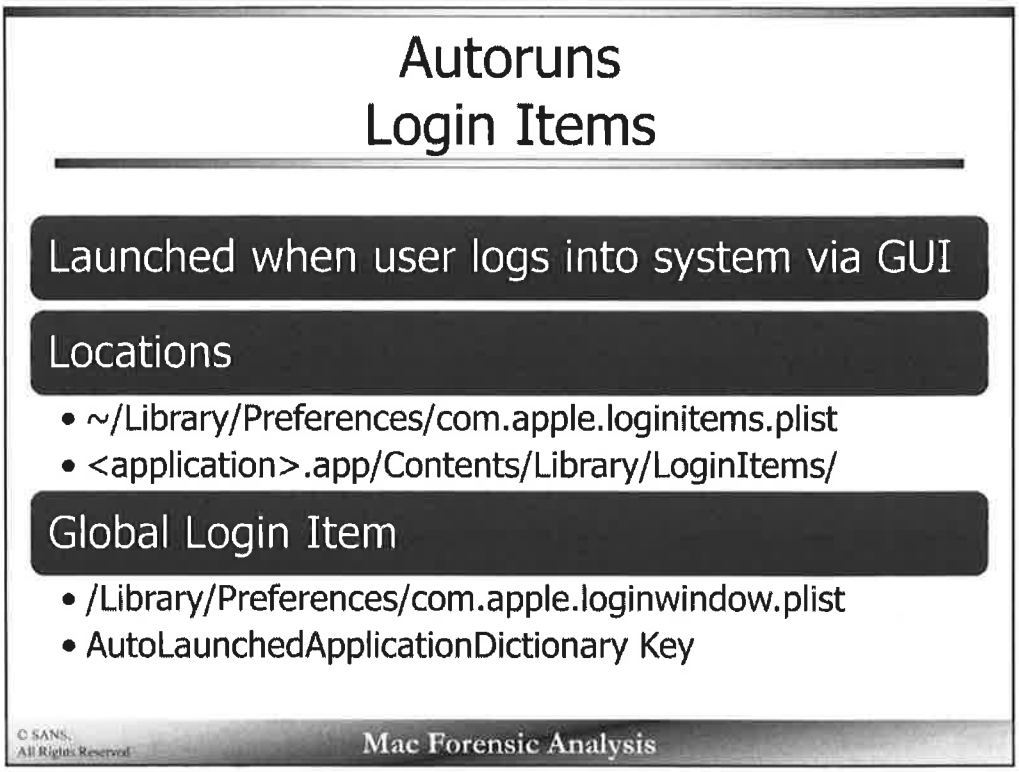
Good examples of launch daemons are the Unix periodic maintenance scripts; daily, weekly, and monthly.

The screenshot shows the output from the daily script in the `daily.out` file. The output in the log has a timestamp of April 13th at 03:15:00 EDT 2012. 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 `StartCalendarInterval` which starts the daemon at a specified time. This key shows the hour and minute of the day that this script is intended to run, 03:15.

```
Fri Apr 13 03:15:00 EDT 2012

Removing old temporary files:

Cleaning out old system announcements:

Removing stale files from /var/rwho:

Removing scratch fax files

Disk status:
Filesystem                          Size   Used  Avail Capacity  Mounted on
/dev/disk1                         698Gi  431Gi  267Gi    62%    /
localhost:/YNU-3NIWZFYxgBrbEqggLJ  698Gi  698Gi    0Bi   100%    /Volumes/MobileBackups

Network interface status:
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16384 <Link#1>                        38376     0    38376     0     0
lo0   16384 localhost   fe80:1::1           38376     -    38376     -     -
lo0   16384 127           localhost         38376     -    38376     -     -
lo0   16384 ip6-localho ::1                 38376     -    38376     -     -
gif0* 1280  <Link#2>                            0     0        0     0     0
stf0* 1280  <Link#3>                            0     0        0     0     0
en0   1500  <Link#4>    c4:2c:03:09:ca:fd       0     0        0     0     0
en1   1500  <Link#5>    90:27:e4:f8:e6:5f 37611065     0 105742931     0     0
en1   1500  bit.local   fe80:5::9227:e4ff 37611065     - 105742931     -     -
en1   1500  192.168.1     bit              37611065     - 105742931     -     -
fw0   4078  <Link#6>    e8:06:88:ff:fe:d5:5d:08       0     0        0     0     0
p2p0  2304  <Link#7>    02:27:e4:f8:e6:5f       0     0        0     0     0
vmnet 1500  <Link#8>    00:50:56:c0:00:01       0     0        0     0     0
vmnet 1500  172.16.73/24  172.16.73.1           0     -        0     -     -
vmnet 1500  <Link#9>    00:50:56:c0:00:08       0     0        0     0     0
vmnet 1500  192.168.158   192.168.158.1         0     -        0     -     -

Local system status:
 3:15  up 3 days,  8:06, 4 users, load averages: 0.55 0.57 0.56

-- End of daily output --
```

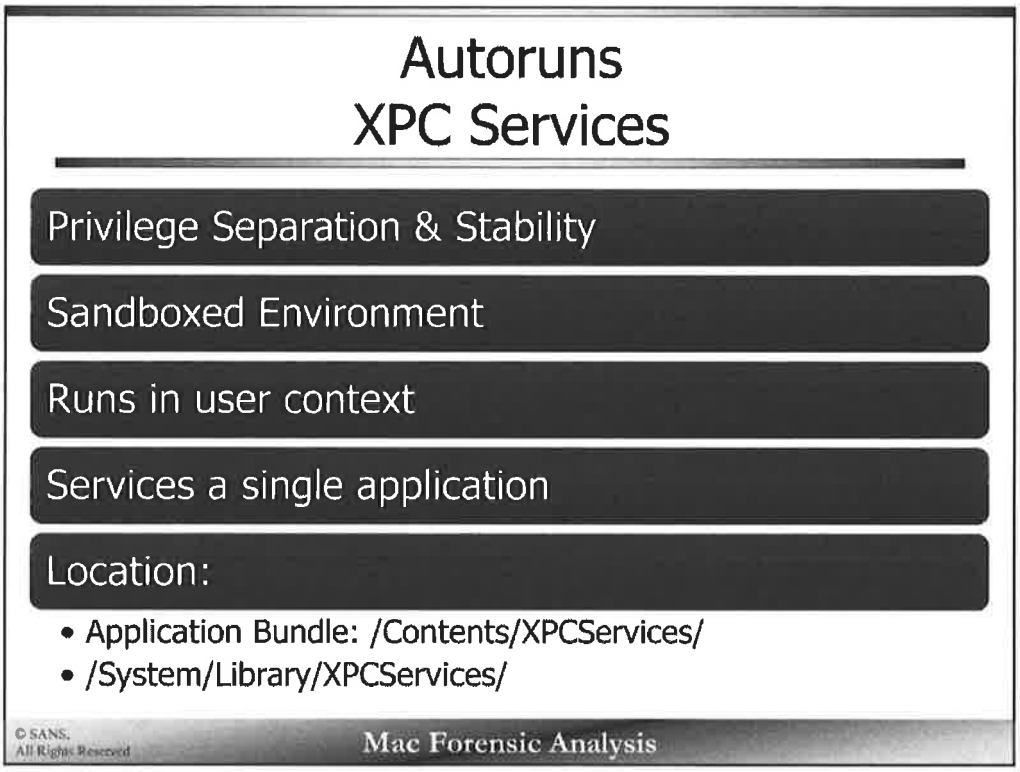
Key	Type	Value
Label	String	com.apple.periodic-daily
▼ ProgramArguments	Array	(2 items)
Item 0	String	/usr/sbin/periodic
Item 1	String	daily
LowPriorityIO	Boolean	YES
Nice	Number	1
▼ StartCalendarInterval	Diction...	(2 items)
Hour	Number	3
Minute	Number	15
AbandonProcessGroup	Boolean	YES



Previously mentioned in Section 2, login items are launched when a user logs into the system using the GUI. 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.

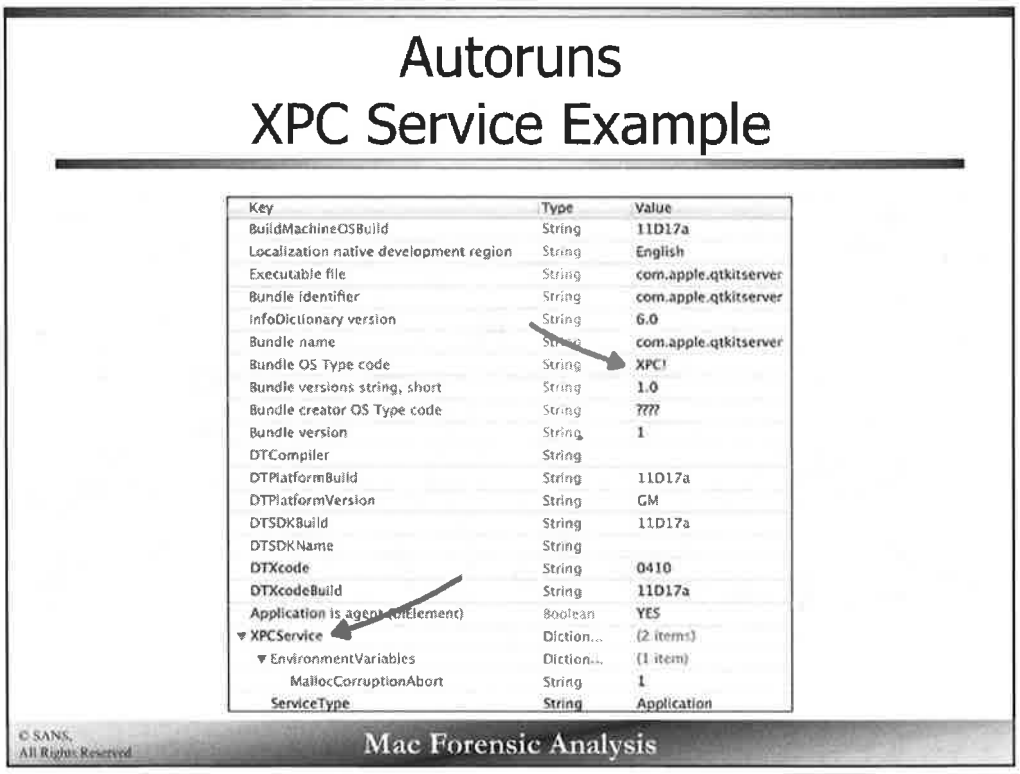


XPC (Interprocess Communication) services are used to control the stability of a system. These services will only service one application. If the service crashes, only the application will crash rather than the whole system. These services run in the user context and are located in various `XPCServices`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

References:

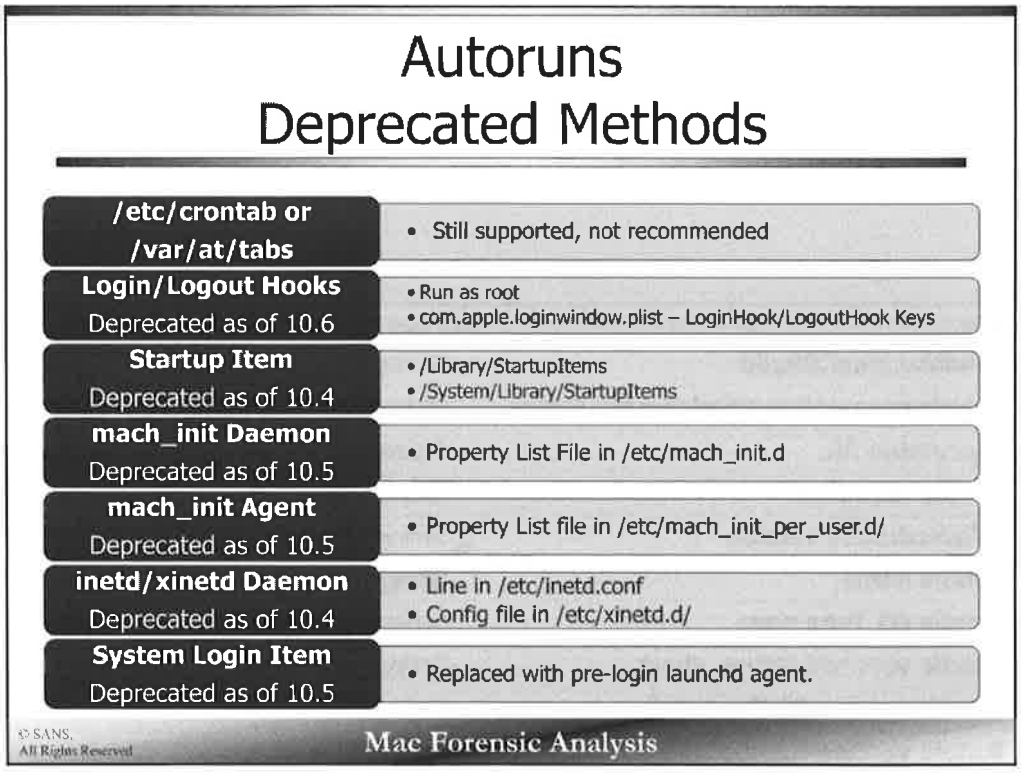
Creating XPC Services - Apple Developer Documentation

http://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/BPSystemStartup/Chapters/CreatingXPCServices.html



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



There are quite a few deprecated methods for starting applications on Mac OS X systems.

The Unix standby `/etc/crontab` can still be implemented, however the previously mentioned methods are preferred. The

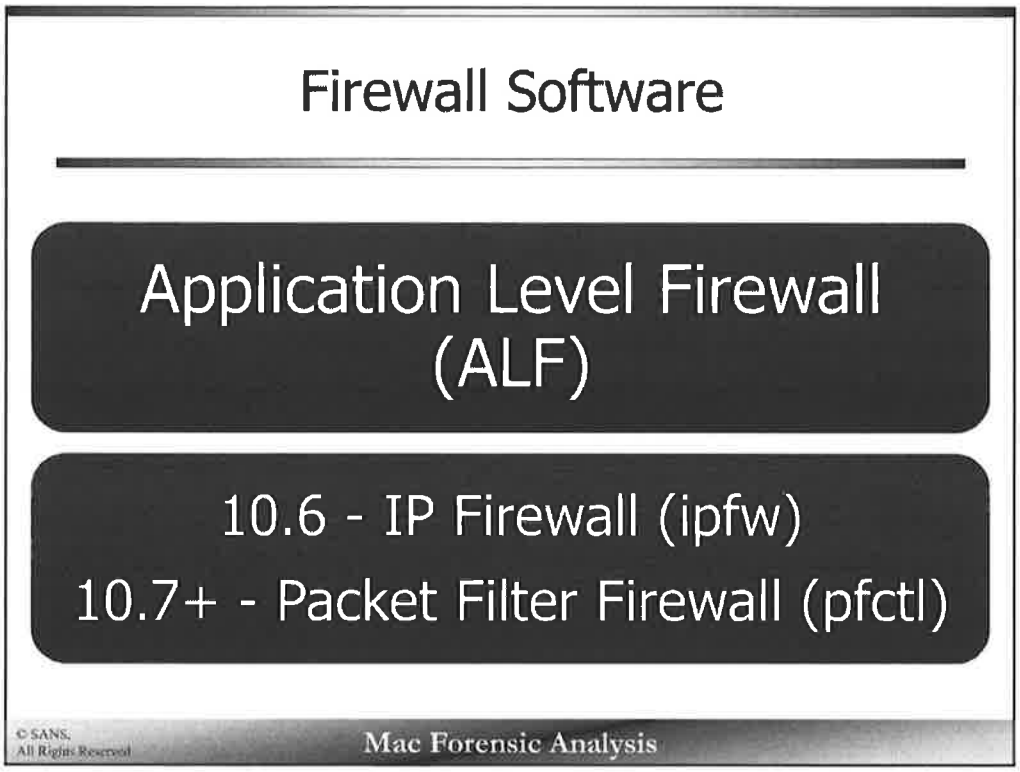
Login and logout hooks are located in `com.apple.loginwindow.plist` property list as keys of the same name.

Startup Items has been reinvented as Login Items; however, they may still be found in various `StartupItems` directories.

`mach_init` agents and daemons were once located in the `/etc` directory as property lists.

`inetd/xinetd` daemons may be found in the `/etc` directory in the `inetd.conf` file or in the `xinetd.d/` directory.

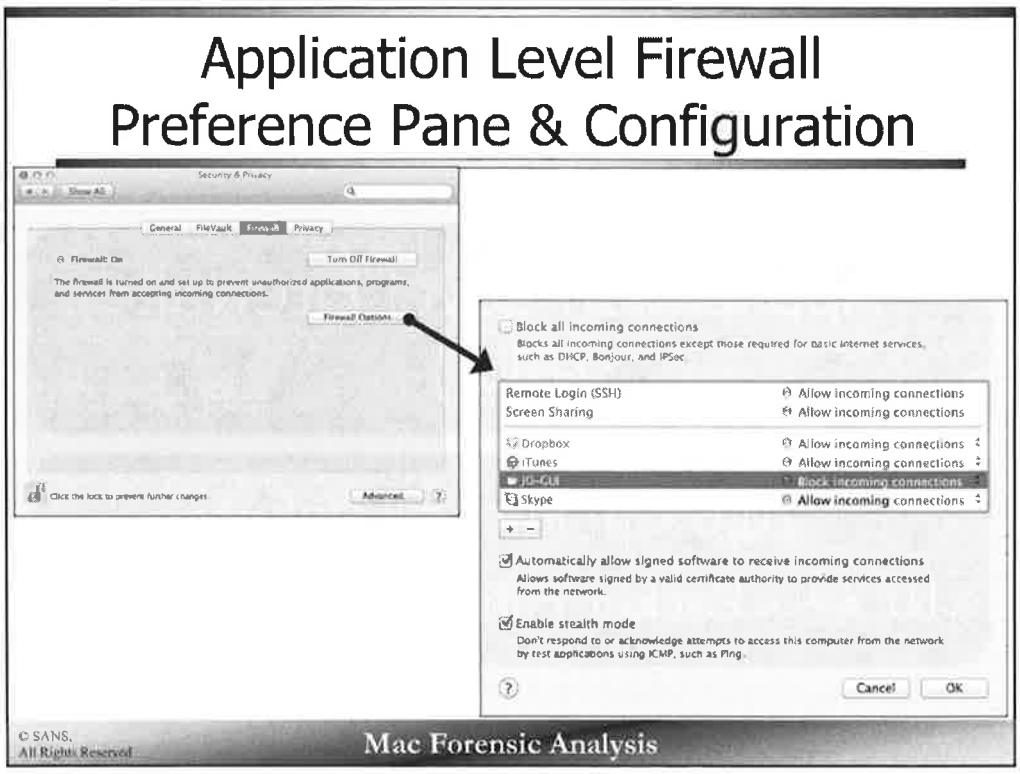
The `System Login Items` were replaced with pre-login launch agents, however recall that it is still possible to configure a global login item in the `com.apple.loginwindow.plist` file in the `/Library/Preferences` directory.



Mac OS X systems come with two firewalls. An application level firewall (host-based application firewall) and an IP/Packet Filtering firewall.

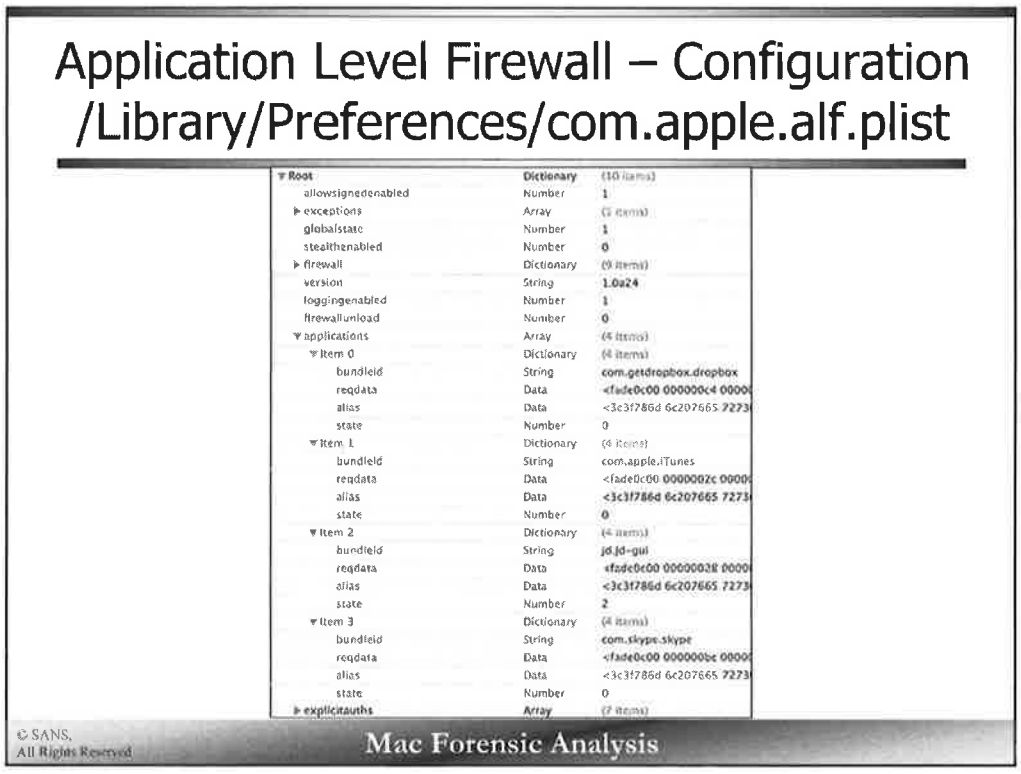
A non-savvy user may use the application level firewall via the GUI, while a more technical user may use the IP firewall via the `ipfw` or `pfctl` command.

If used, the Packet Filtering Firewall configuration can be found in `/etc/pf.conf`.



The default firewall program on Mac OS X 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 right shows the `Firewall Options` which show the remote access options and specific application configurations.



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

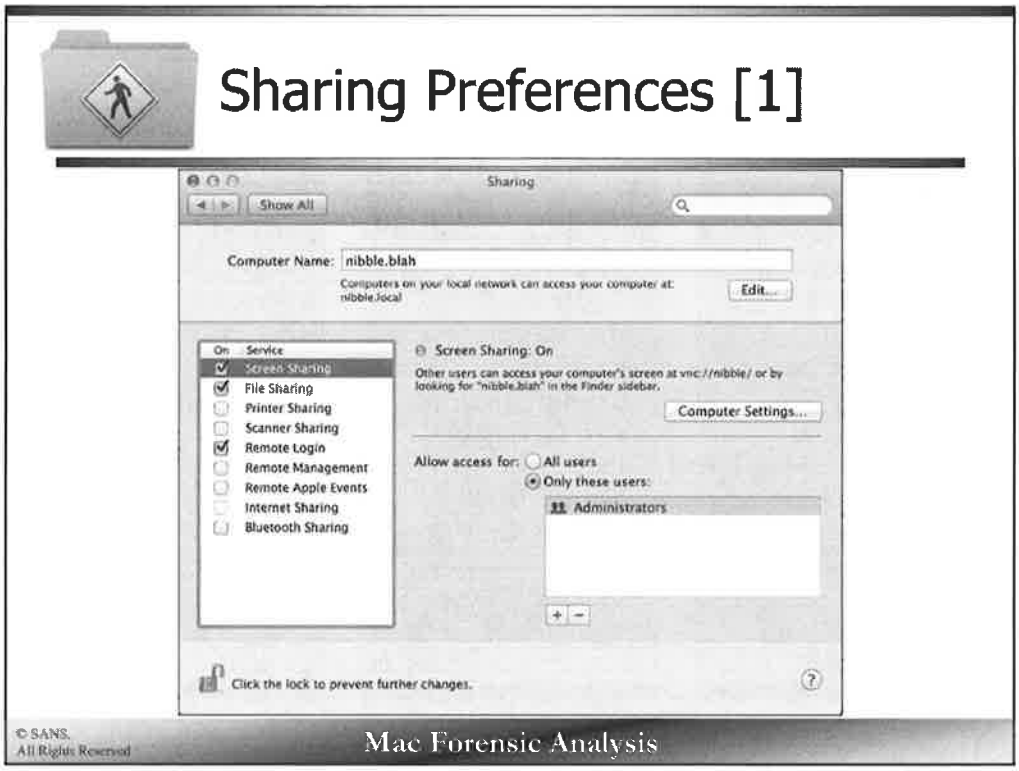
In the previous screenshot of the application firewall, there are two checkboxes the users can configure. The following keys hold this configuration:

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

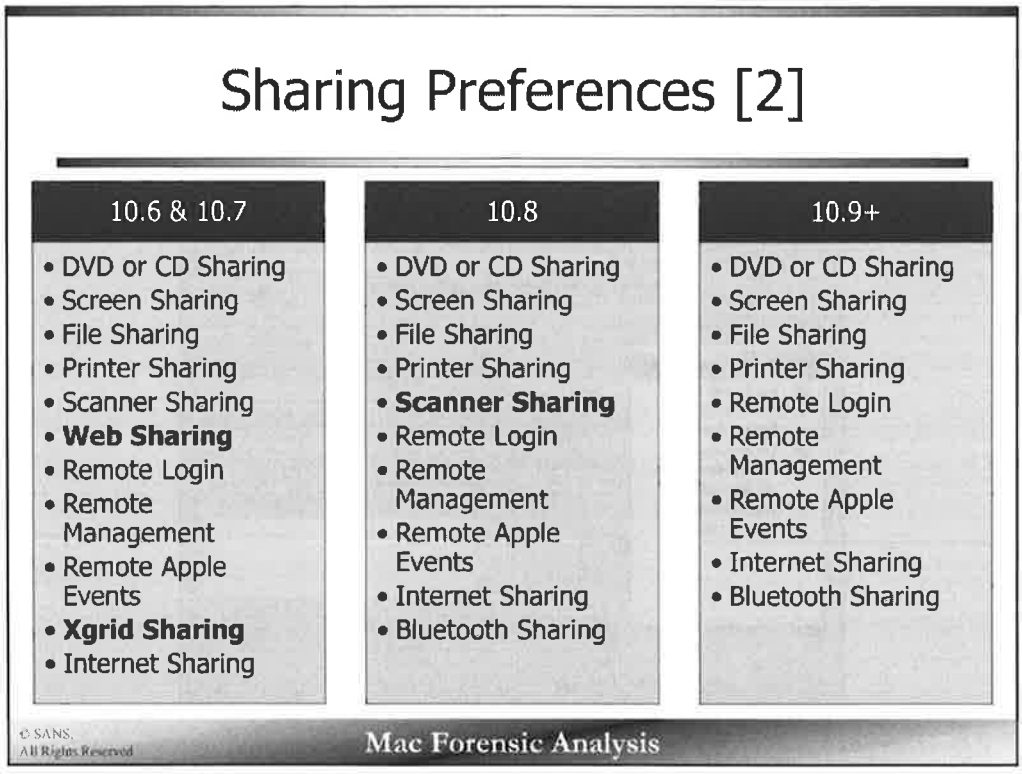
The `applications` key lists the applications configured in the firewall as shown in the screenshot on the previous slide. Each application item contains the applications bundle ID, alias data and the state. A state of "0" allows incoming connections, while a state of "2" blocks incoming connections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

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

▼ Root	Dictionary	(10 items)
allowsignedenabled	Number	1
▶ exceptions	Array	(5 items)
globalstate	Number	1
stealthenabled	Number	0
▶ firewall	Dictionary	(9 items)
version	String	1.0a24
loggingenabled	Number	1
firewallunload	Number	0
▼ applications	Array	(4 items)
▼ Item 0	Dictionary	(4 items)
bundleid	String	com.getdropbox.dropbox
reqdata	Data	<fade0c00 000000c4 0000
alias	Data	<3c3f786d 6c207665 7273
state	Number	0
▼ Item 1	Dictionary	(4 items)
bundleid	String	com.apple.iTunes
reqdata	Data	<fade0c00 0000002c 0000
alias	Data	<3c3f786d 6c207665 7273
state	Number	0
▼ Item 2	Dictionary	(4 items)
bundleid	String	jd.jd-gui
reqdata	Data	<fade0c00 00000028 0000
alias	Data	<3c3f786d 6c207665 7273
state	Number	2
▼ Item 3	Dictionary	(4 items)
bundleid	String	com.skype.skype
reqdata	Data	<fade0c00 000000bc 0000
alias	Data	<3c3f786d 6c207665 7273
state	Number	0
▶ explicitauths	Array	(7 items)

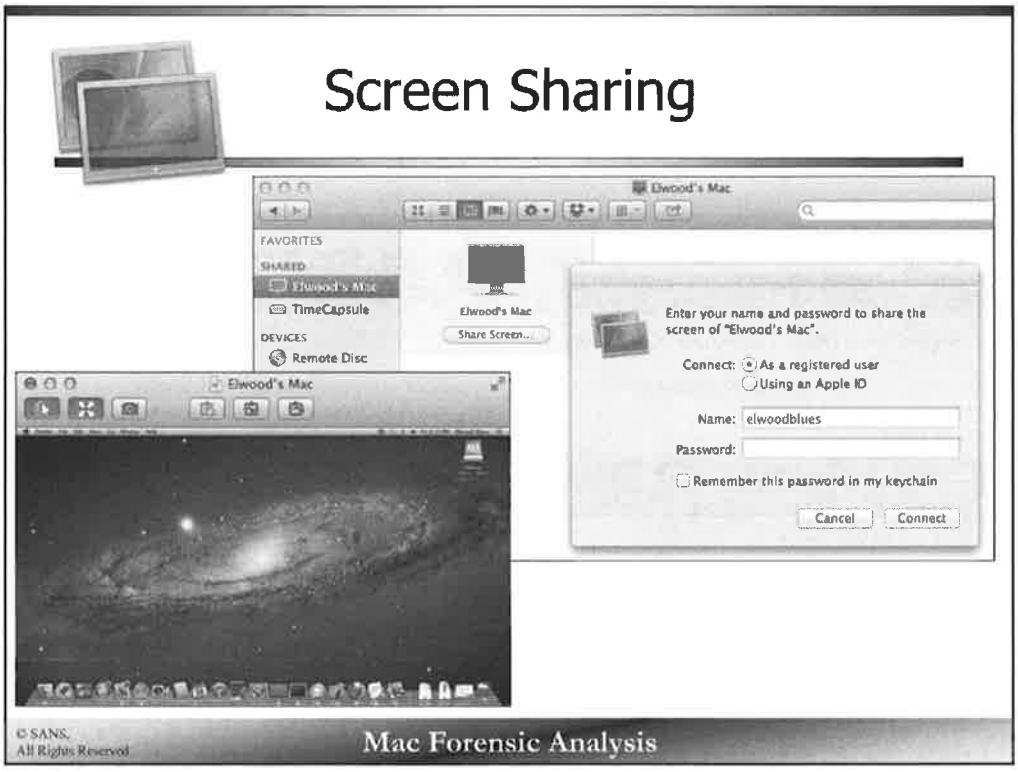


The `Sharing` preferences pane shown in the screenshot above contains the items that can be shared from the system. By default none of these are enabled.

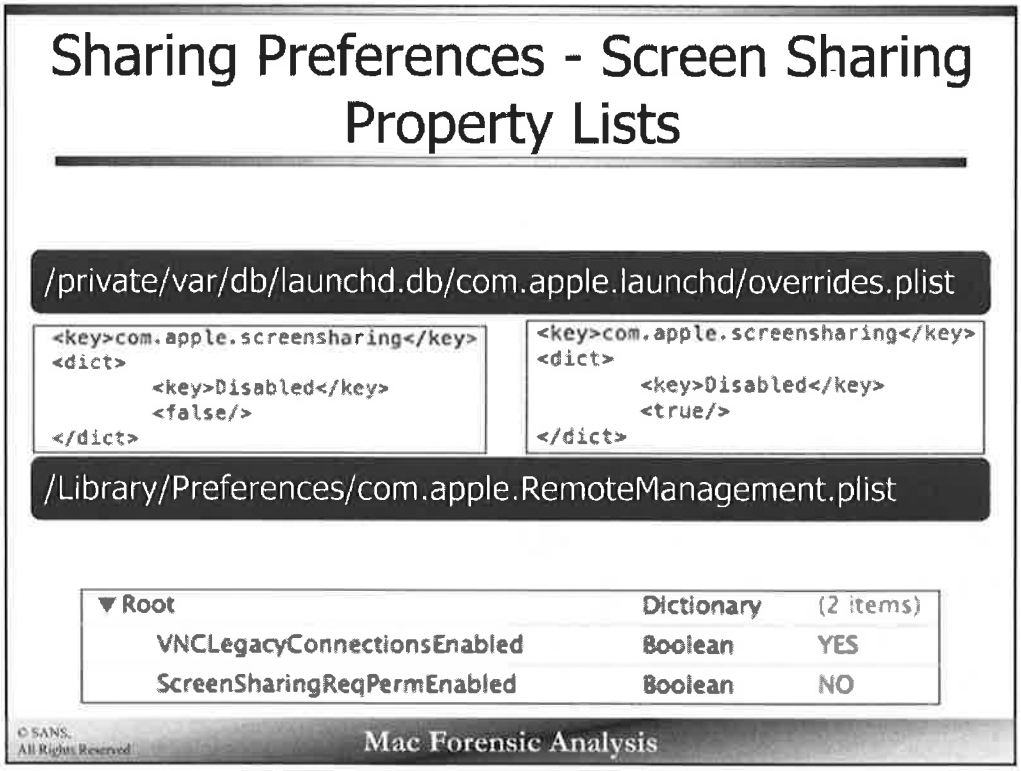


The shared preferences options have kept changing from each OS X version. The changes bolded are no longer available on the next OS X version due to discontinued hardware and software (DVD/CD drive, Xgrid) or just limiting user issues (Web Sharing can still be enabled via the command line.)

It is worth noting the DVD or CD Sharing option depends on the hardware of the system. If the system does not have a CD/DVD drive to share, the option will not be listed.



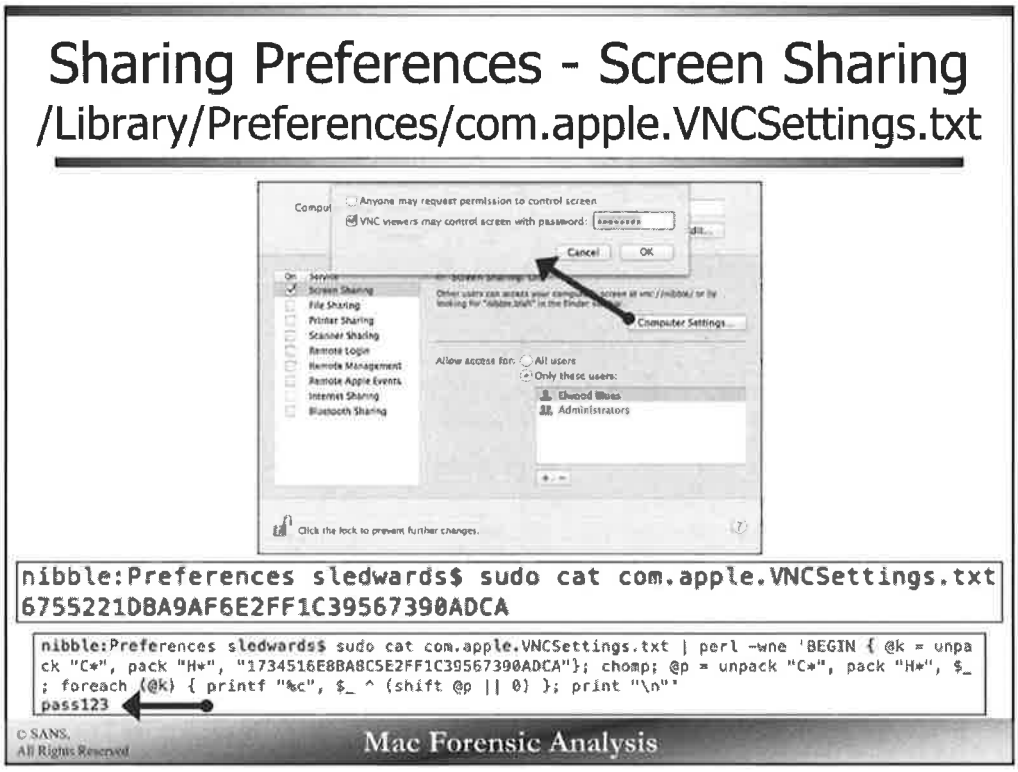
The Screen Sharing application, located in `/System/Library/CoreServices/` rather than `/Applications`, is used to connect to other systems using the VNC protocol.



The `overrides.plist` file contains many configuration keys for various sharing settings. The `com.apple.screensharing` key contains the `Disabled` subkey. The value for this subkey can either be `true` or `false`. If screen sharing is enabled, the disabled key is "`false`", while if screen sharing is disabled, the key will be "`true`".

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.

10.10 does not appear to use the `overrides.plist`, more research is need to determine where this information is stored.



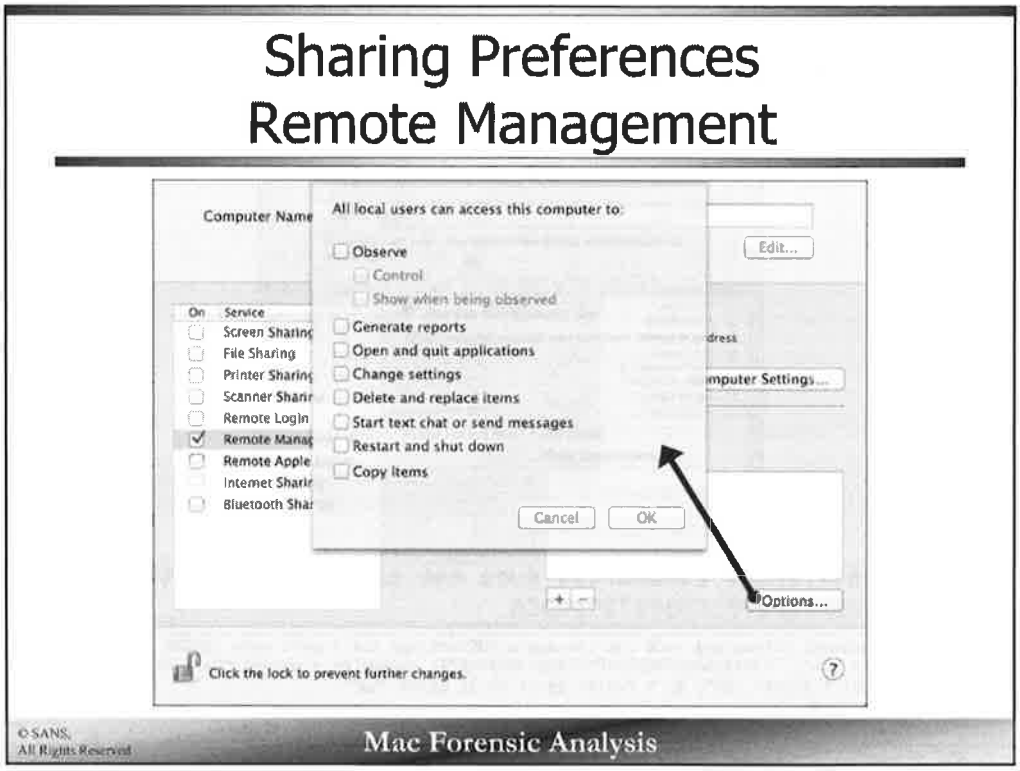
The `com.apple.VNCSettings.txt` text file contains the XOR'ed "encrypted" password to access the system via a VNC viewer client. We can use the Perl script created by Ben Low (http://lists.apple.com/archives/remote-desktop/2005/Oct/msg00026.html) to "decrypt" the VNC password using the XOR key "`1734516E8BA8C5E2FF1C39567390ADCA`".

```
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"'
```

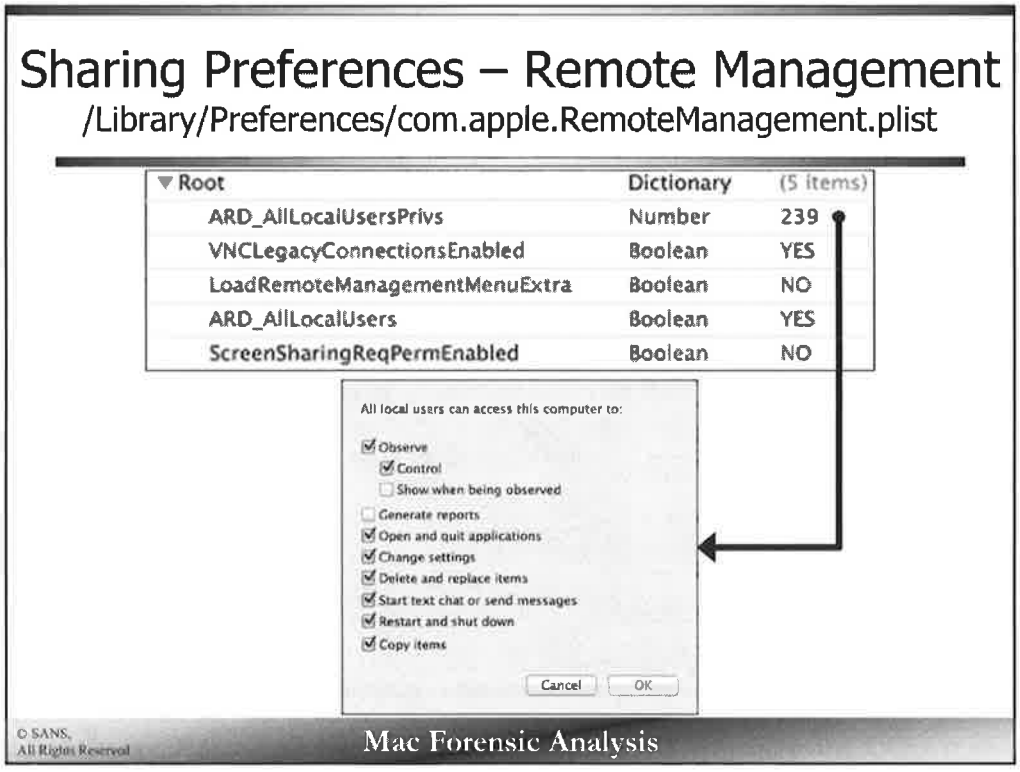
References:

VNC Password - command line configuration of VNC

http://lists.apple.com/archives/remote-desktop/2005/Oct/msg00026.html

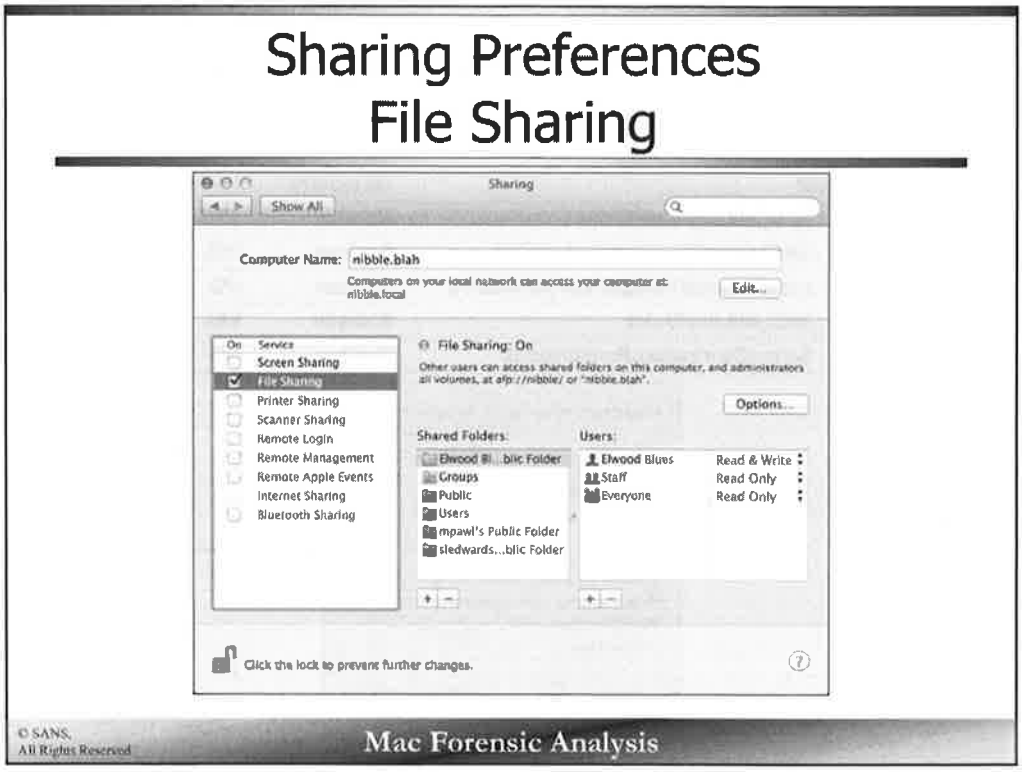


Remote Management is inclusive of Screen Sharing. When the Remote Management box is checked, the Screen Sharing box will be grayed out. Shown in the screenshot above, when Remote Management is checked, a box will pop up allowing the user to choose the functions available to remotely manage.

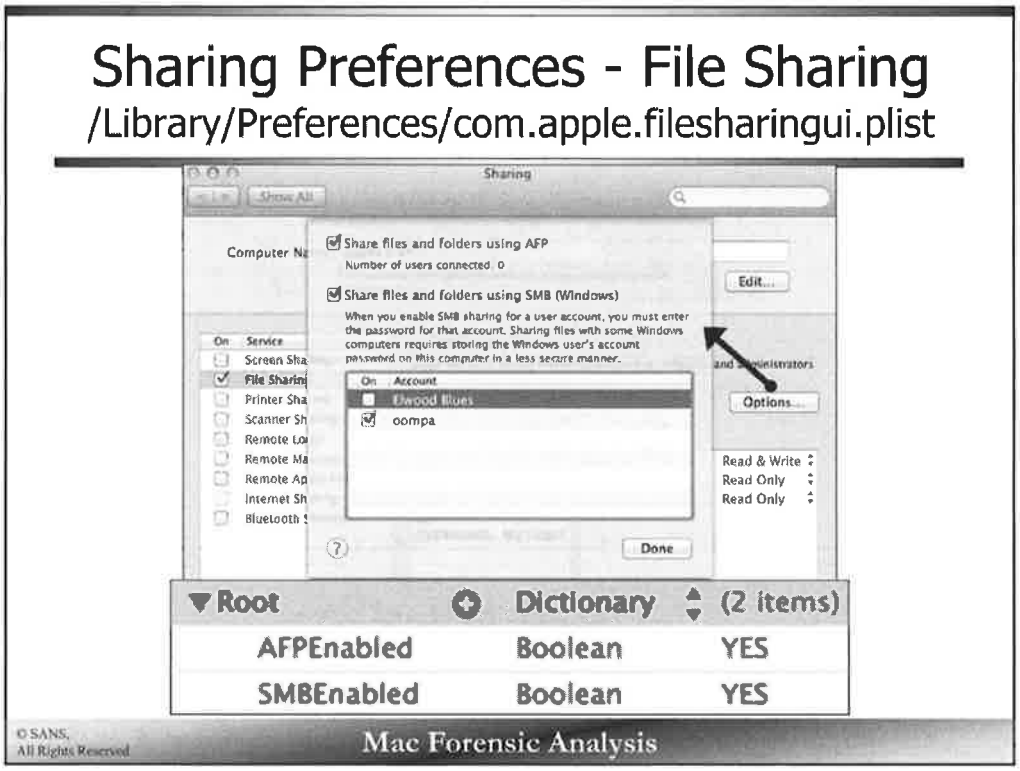


The `com.apple.RemoteManagement.plist` is created in the `/Library/Preferences/` directory when the Screen Sharing or Remote Management options are selected in the Sharing preferences pane.

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.



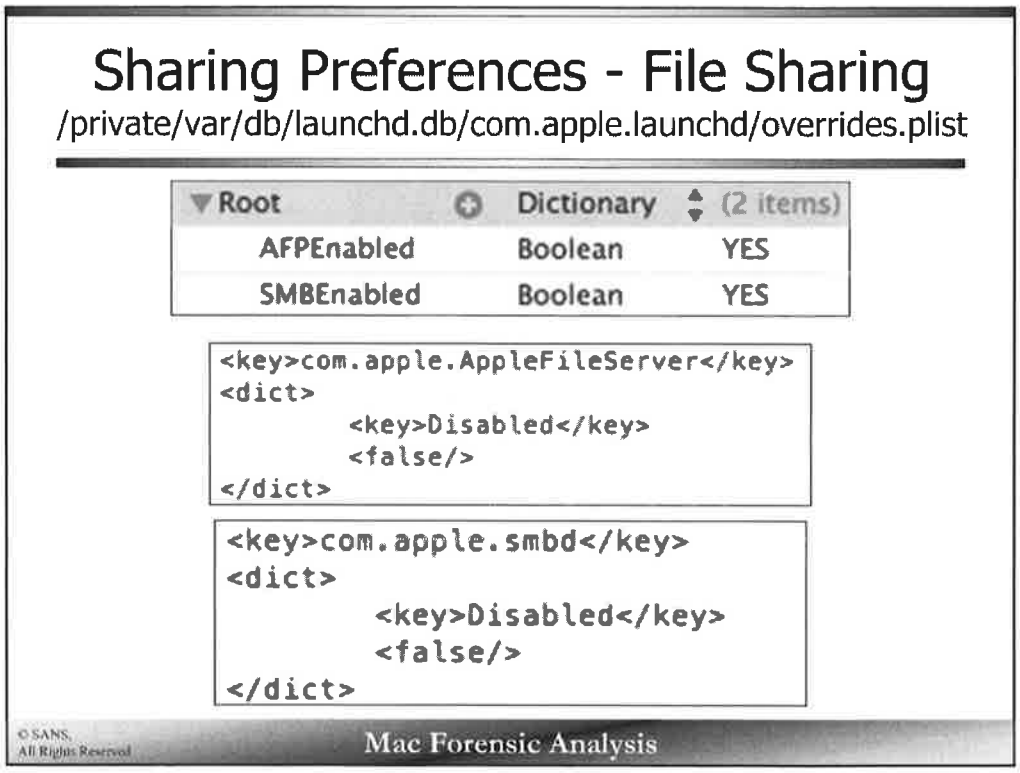
File Sharing allows users to share files on the network or with other users on the system.



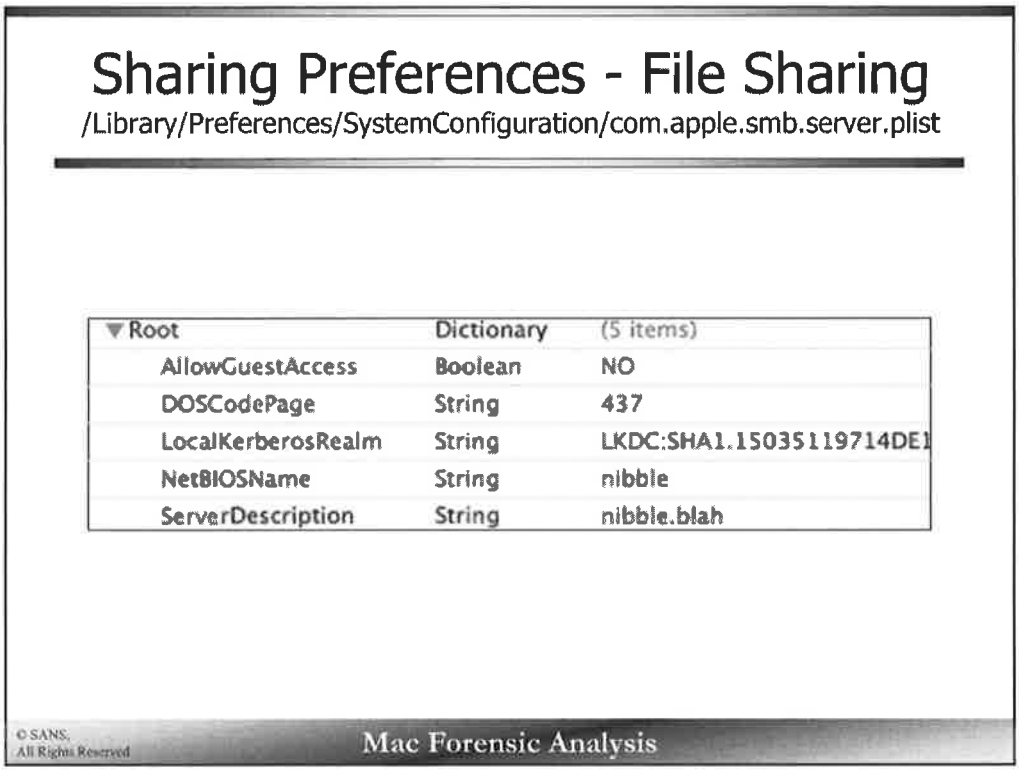
The file sharing preferences file, `com.apple.filesharingui.plist` in the `/Library/Preferences/` directory contains two keys each showing a `YES` or `NO` value depending if the sharing service is enabled.

- `AFPEnabled` – (Apple Filing Protocol) If value is YES, AFP sharing is enabled
- `SMBEnabled` – (Server Message Block) If value is YES, SMB sharing is enabled

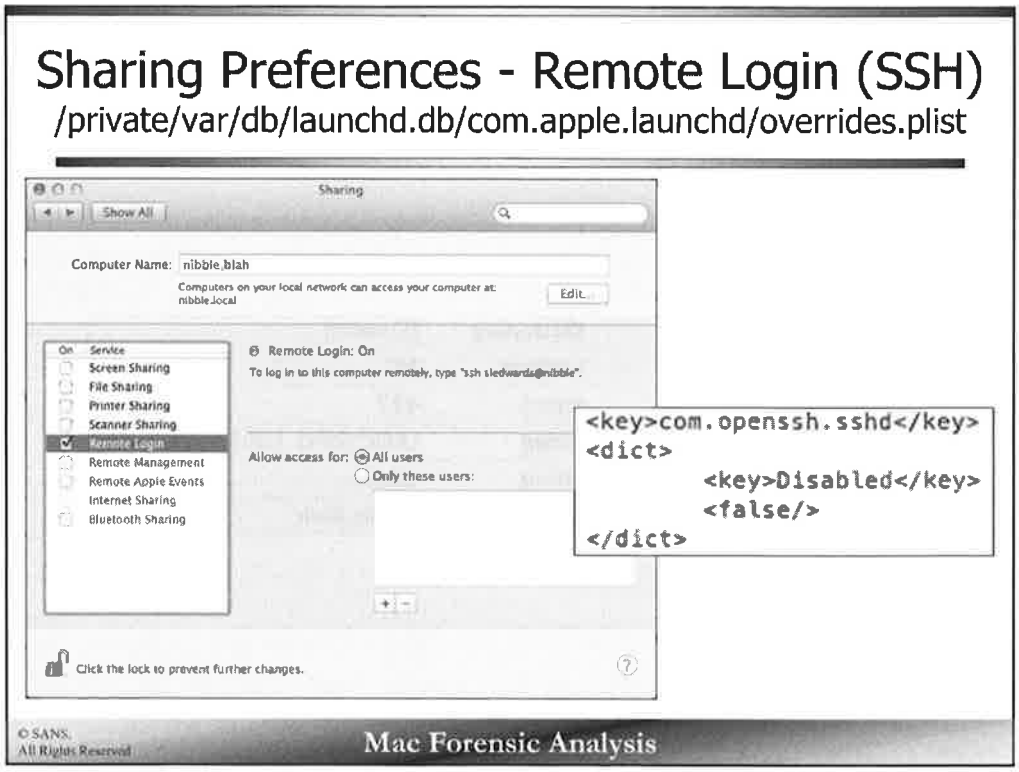
On 10.9 systems, this property list does not appear to be used – instead it only uses the `overrides.plist` described on the next slide.



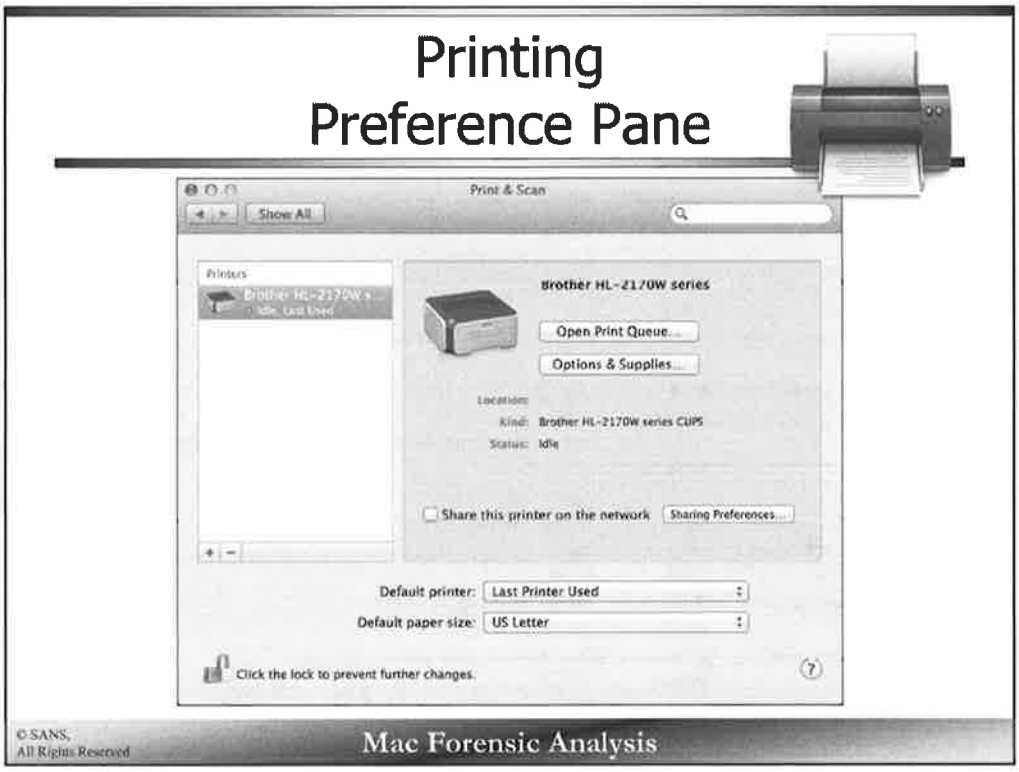
The `overrides.plist` property list located in the `/private/var/db/launchd.db/com.apple.launchd/` directory also contains keys that show if AFP or SMB sharing are enabled. In the screenshot above the two "`Disabled`" keys show false, meaning they are both enabled.



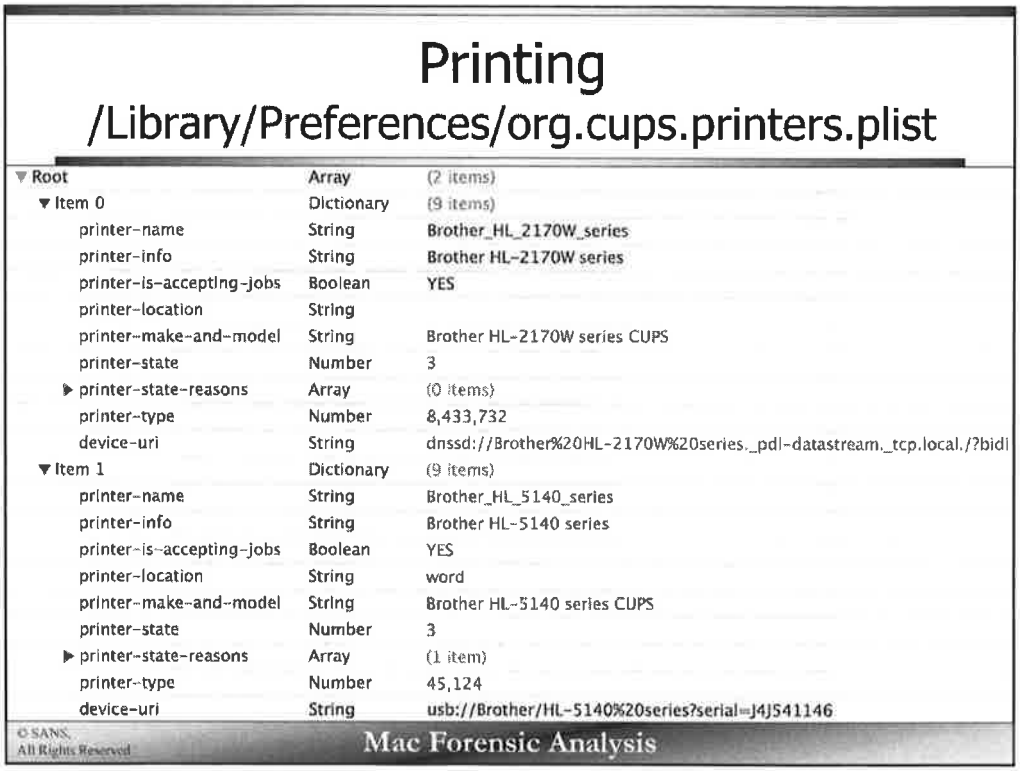
The `com.apple.smb.server.plist` file located in `/Library/Preferences/SystemConfiguration/` contains information specific to SMB sharing. The property list contains the NetBIOS name for the local system.



Remote Login allows a user to remotely login to the system via the SSH (Secure Shell) protocol. The `overrides.plist` file located in the `/private/var/db/launchd.db/com.apple.launchd/` directory contains the key `com.openssh.sshd`. When this key is false, SSH is enabled.



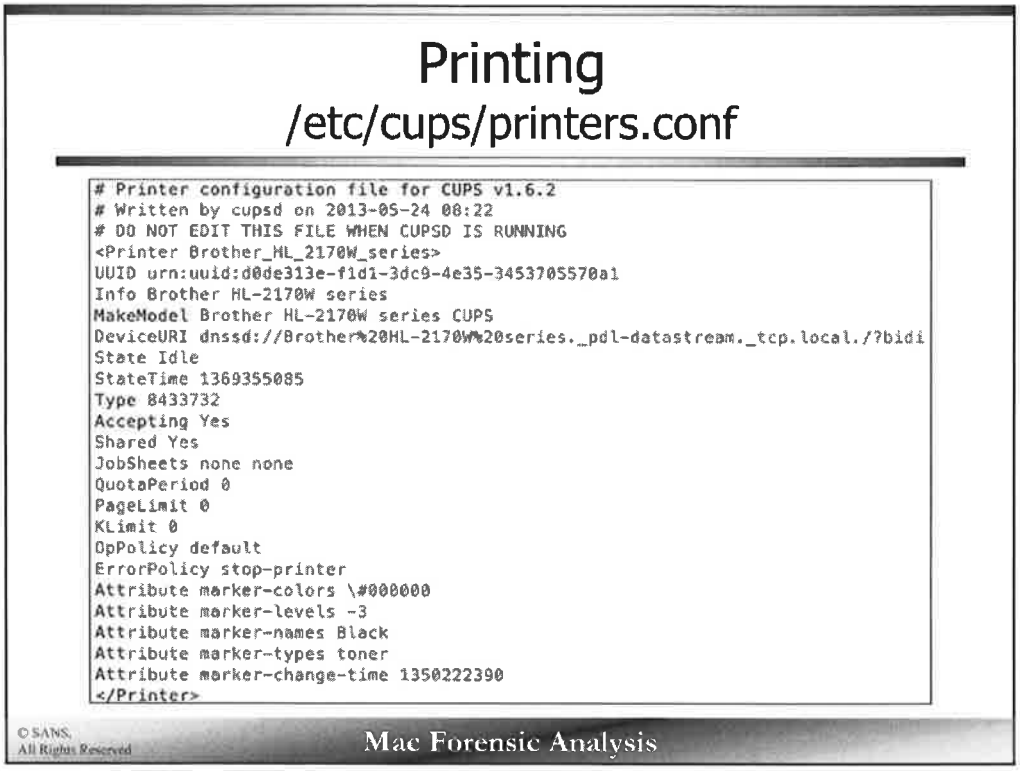
The Printing preference pane contains printer settings for the system. The screenshot shows one printer, a Brother HL-2170W, setup on the system.



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.

▼Root	Array	(1 item)
▼Item 0	Dictionary	(9 items)
printer-name	String	Brother_HL_2170W_series
printer-info	String	Brother HL-2170W series
printer-is-accepting-	Boolean	YES
printer-location	String	
printer-make-and-	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

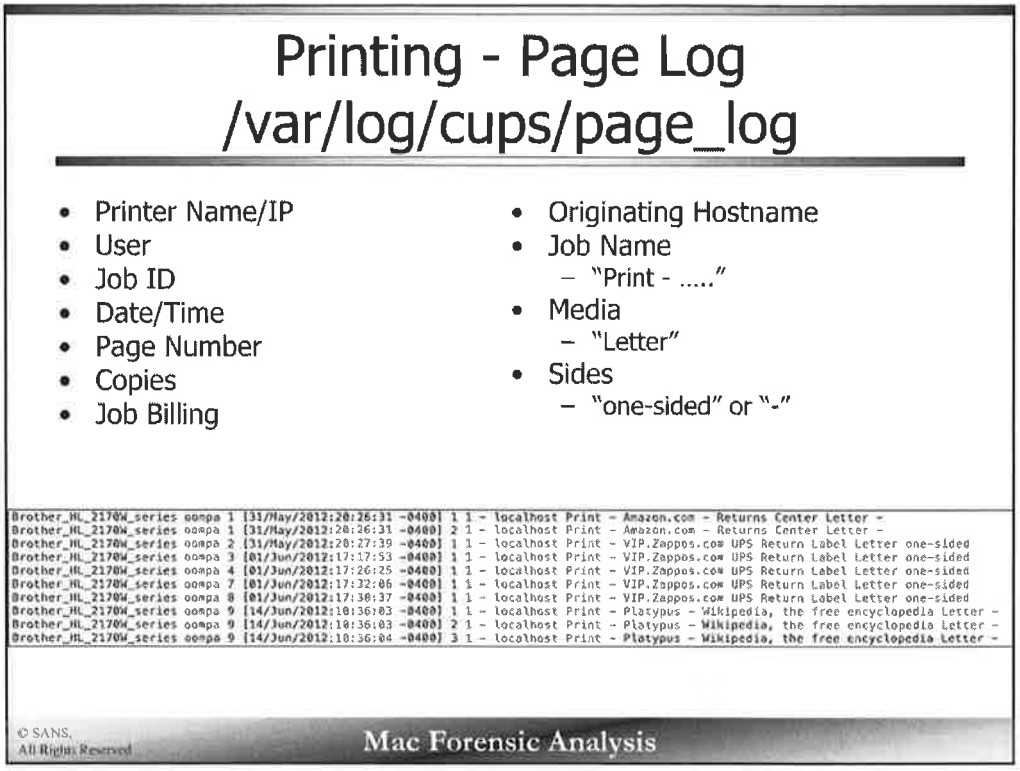


The `printers.conf` file located in `/etc/cups/` contains specific printer information for each printer installed on the system. The screenshot shows the Brother printer is shared amongst other system users ("`Shared Yes`").

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>
```



OS X systems use the Common Unix Printing System (CUPS) to perform printing functions. Each print job is recorded in a few different logs.

The `page_log` located in the `/var/log/cups` directory contains the print job log. Each print job contains metadata listed above, along with the job name which is usually the name of the document or webpage that was printed.

Printer Name/IP – Name or IP of the printer used

User – User Account

Job ID – Incremental print job ID number (starts at 1)

Date/Time – When the job was printed

Page Number – Current page number of print job, each page of a document will have a separate record

Copies – Number of copies of page are printed

Job Billing – Default is dash (-), job-billing attribute if given

Originating Hostname – Hostname of system, although in my experience it is always "localhost"

Job Name – The name of the document, webpage, etc. that is being printed. "Print – " is prepended.

Media – Print media, in the example "Letter" paper was used.

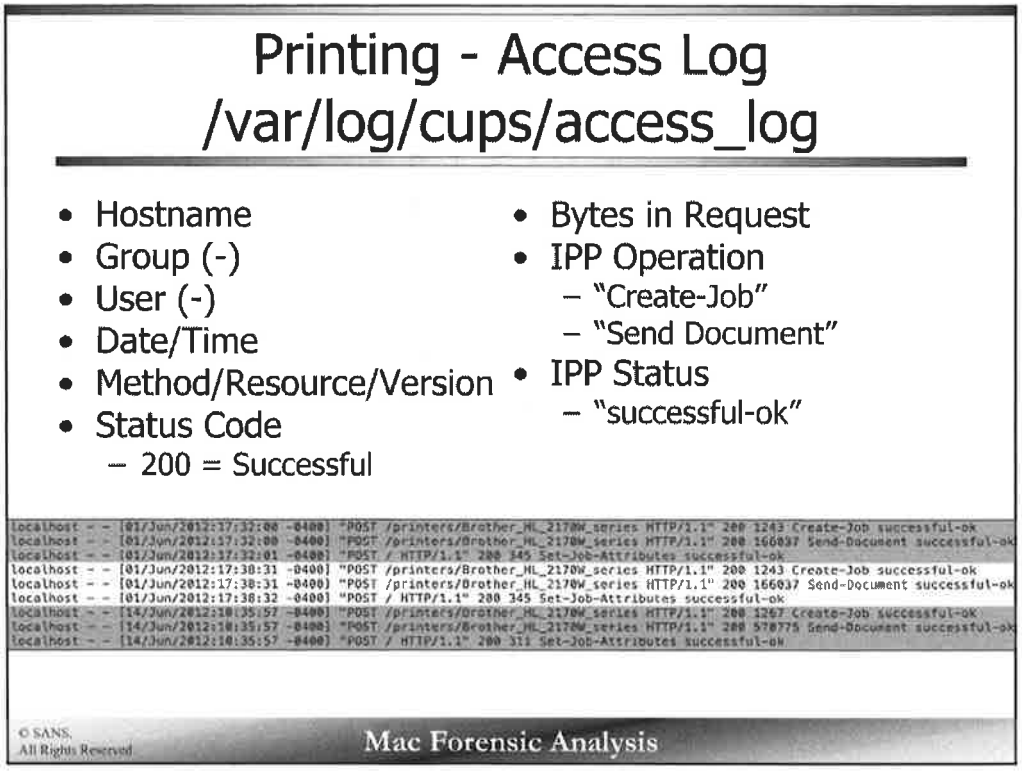
Sides – Printing can occur on one-sided or double-sided. Dash (-) is default.

References:

CUPS Documentation – page_log

http://www.cups.org/documentation.php/ref-page_log.html

```
Brother_HL_2170W_series oompa 1 [31/May/2012:20:26:31 -0400] 1 1 - localhost Print - Amazon.com - Returns Center Letter -
Brother_HL_2170W_series oompa 1 [31/May/2012:20:26:31 -0400] 2 1 - localhost Print - Amazon.com - Returns Center Letter -
Brother_HL_2170W_series oompa 2 [31/May/2012:20:27:39 -0400] 1 1 - localhost Print - VIP.Zappos.com UPS Return Label Letter one-sided
Brother_HL_2170W_series oompa 3 [01/Jun/2012:17:17:53 -0400] 1 1 - localhost Print - VIP.Zappos.com UPS Return Label Letter one-sided
Brother_HL_2170W_series oompa 4 [01/Jun/2012:17:26:25 -0400] 1 1 - localhost Print - VIP.Zappos.com UPS Return Label Letter one-sided
Brother_HL_2170W_series oompa 7 [01/Jun/2012:17:32:06 -0400] 1 1 - localhost Print - VIP.Zappos.com UPS Return Label Letter one-sided
Brother_HL_2170W_series oompa 8 [01/Jun/2012:17:38:37 -0400] 1 1 - localhost Print - VIP.Zappos.com UPS Return Label Letter one-sided
Brother_HL_2170W_series oompa 9 [14/Jun/2012:10:36:03 -0400] 1 1 - localhost Print - Platypus - Wikipedia, the free encyclopedia Letter -
Brother_HL_2170W_series oompa 9 [14/Jun/2012:10:36:03 -0400] 2 1 - localhost Print - Platypus - Wikipedia, the free encyclopedia Letter -
Brother_HL_2170W_series oompa 9 [14/Jun/2012:10:36:04 -0400] 3 1 - localhost Print - Platypus - Wikipedia, the free encyclopedia Letter -
```



The `access_log` in the `/var/log/cups/` directory contains more print job metadata including the size of the job and if the print job was successful or not. These entries can be correlated with the jobs found in the `page_log`.

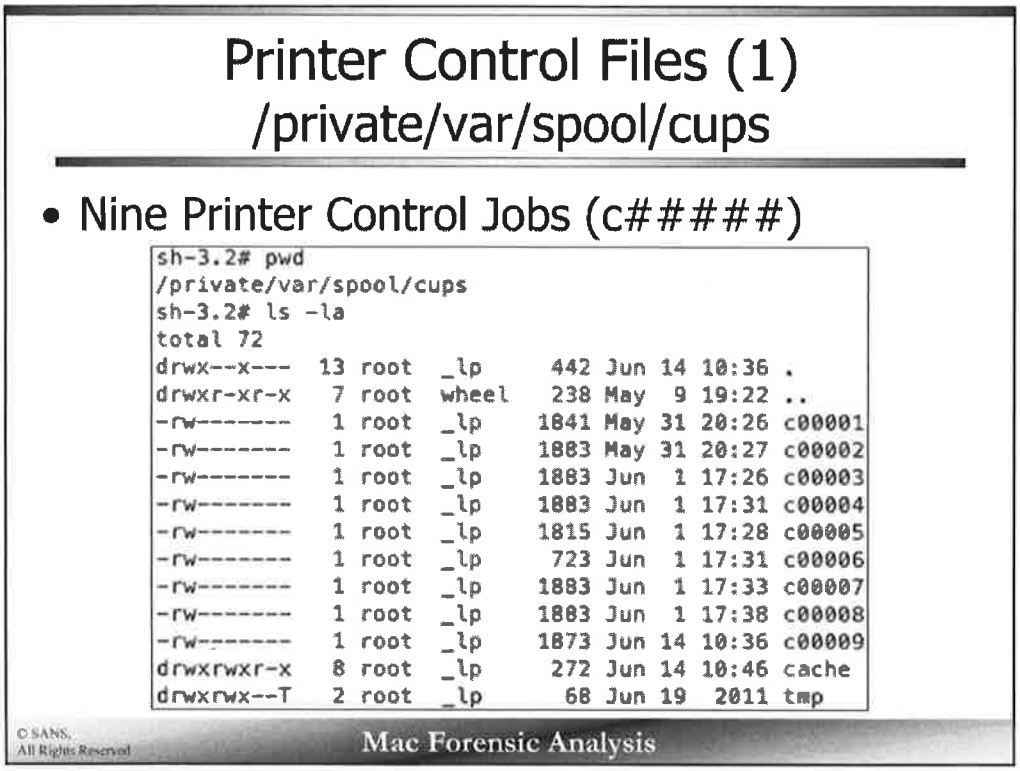
IPP = Internet Printing Protocol

Hostname – System hostname, localhost is used in most cases
Group – Always a dash (-)
User – Always a dash (-), username may be filled in if cupsd.conf was changed
Date/Time – Date and time of print job
Method– HTTP method
Resource – Requested resource
Version – HTTP version
Status Code – HTTP status code, 200 is successful
Bytes – Number of bytes in non-IPP request
IPP Operation – IPP Operation Name
IPP Status – IPP stats response

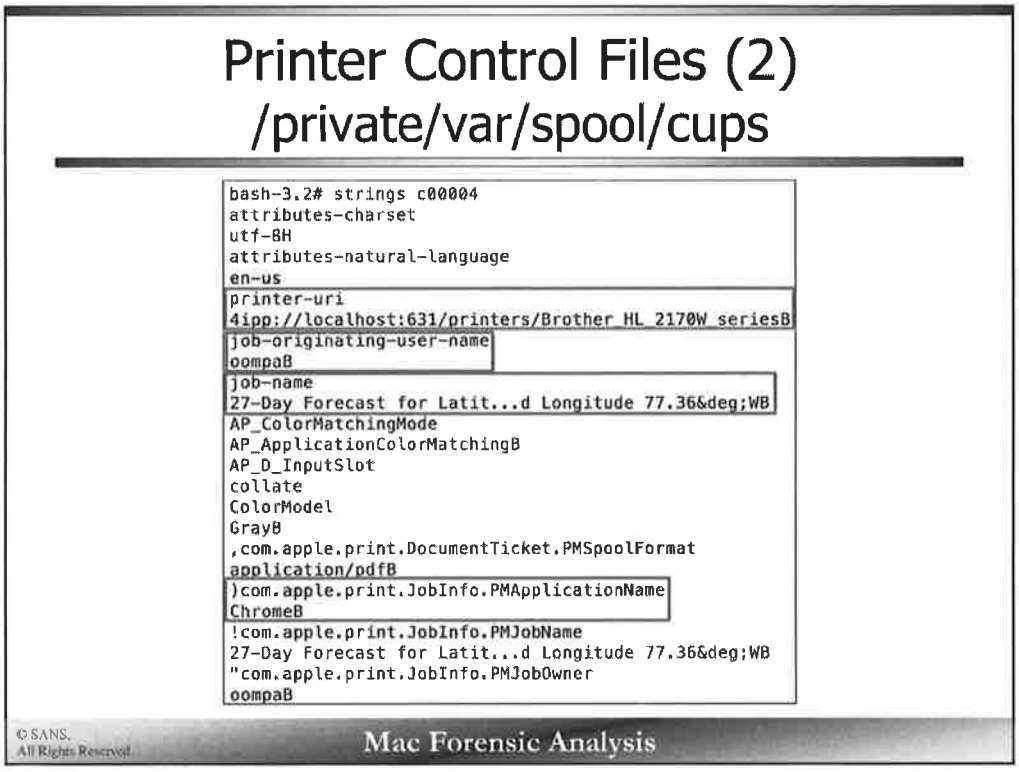
References:
CUPS Documentation – access_log
http://www.cups.org/documentation.php/ref-access_log.html

CUPS Documentation – cupsd
http://www.cups.org/documentation.php/ref-cupsd-conf.html

```
localhost - - [01/Jun/2012:17:32:00 -0400] "POST /printers/Brother_HL_2170W_series HTTP/1.1" 200 1243 Create-Job successful-ok
localhost - - [01/Jun/2012:17:32:00 -0400] "POST /printers/Brother_HL_2170W_series HTTP/1.1" 200 166037 Send-Document successful-ok
localhost - - [01/Jun/2012:17:32:01 -0400] "POST / HTTP/1.1" 200 345 Set-Job-Attributes successful-ok
localhost - - [01/Jun/2012:17:38:31 -0400] "POST /printers/Brother_HL_2170W_series HTTP/1.1" 200 1243 Create-Job successful-ok
localhost - - [01/Jun/2012:17:38:31 -0400] "POST /printers/Brother_HL_2170W_series HTTP/1.1" 200 166037 Send-Document successful-ok
localhost - - [01/Jun/2012:17:38:32 -0400] "POST / HTTP/1.1" 200 345 Set-Job-Attributes successful-ok
localhost - - [14/Jun/2012:10:35:57 -0400] "POST /printers/Brother_HL_2170W_series HTTP/1.1" 200 1267 Create-Job successful-ok
localhost - - [14/Jun/2012:10:35:57 -0400] "POST /printers/Brother_HL_2170W_series HTTP/1.1" 200 570775 Send-Document successful-ok
localhost - - [14/Jun/2012:10:35:57 -0400] "POST / HTTP/1.1" 200 311 Set-Job-Attributes successful-ok
```



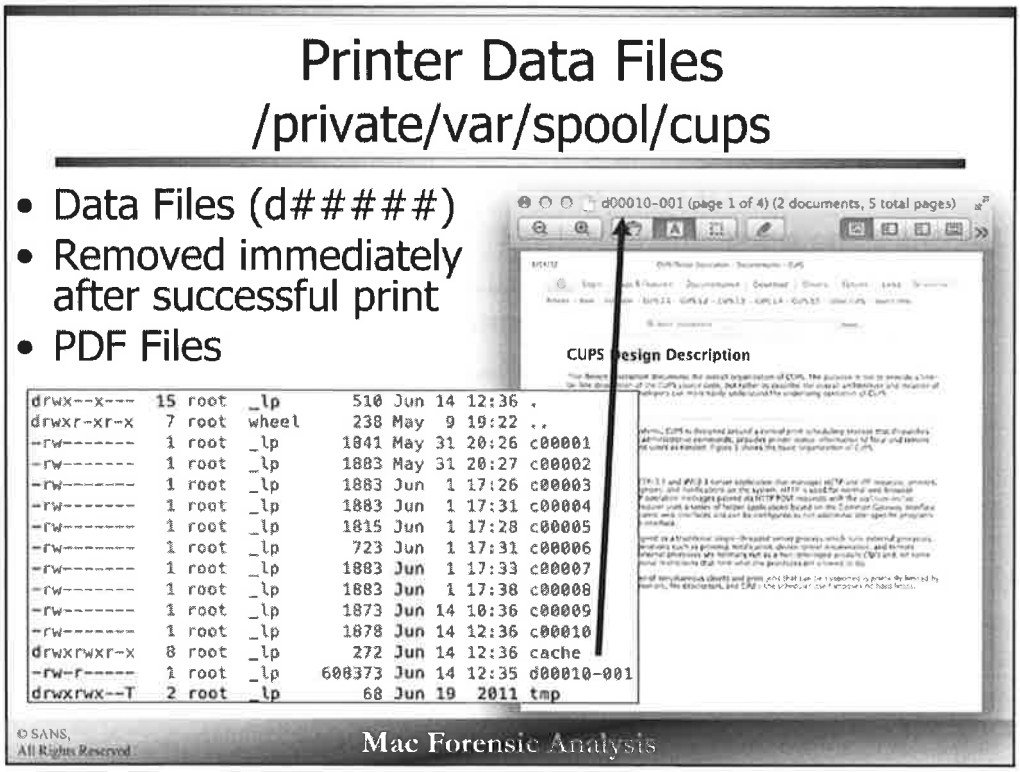
Each print job has a printer control file located in the `/private/var/spool/cups/` directory. Each file labeled `c#####`, correlates with the job ID found in the `page_log`. In the example shown above, the system has printed nine documents (`c00001 - c00009`). Each control file contains print job metadata.



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).

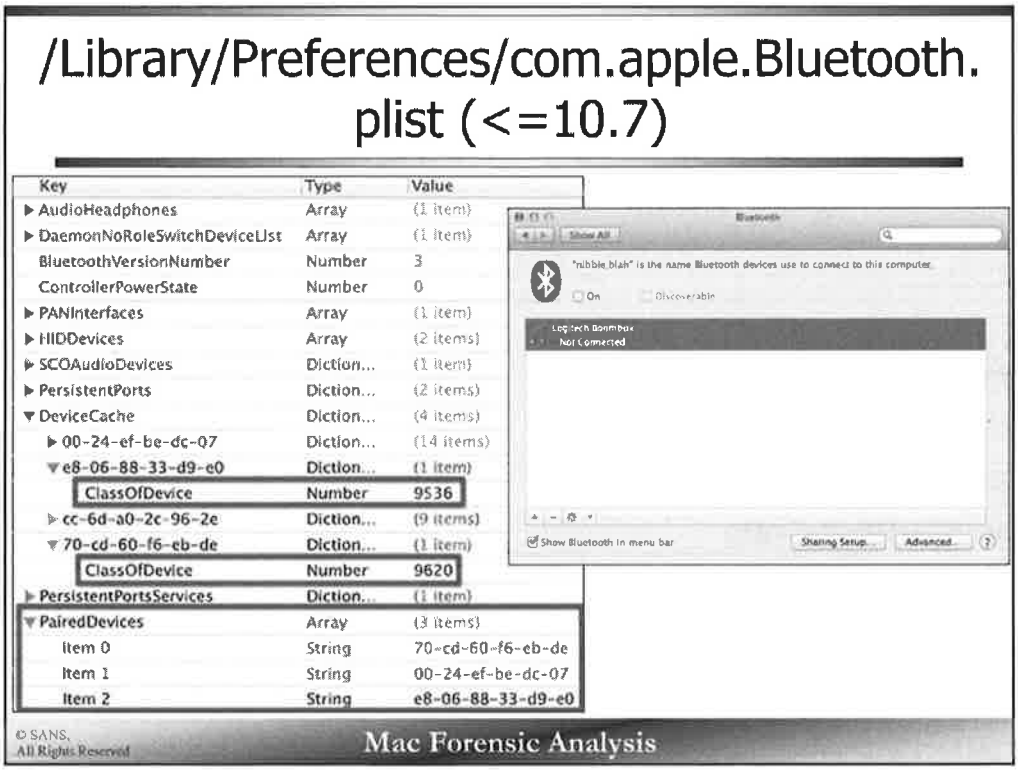
In the example above:

- 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



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 are 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.



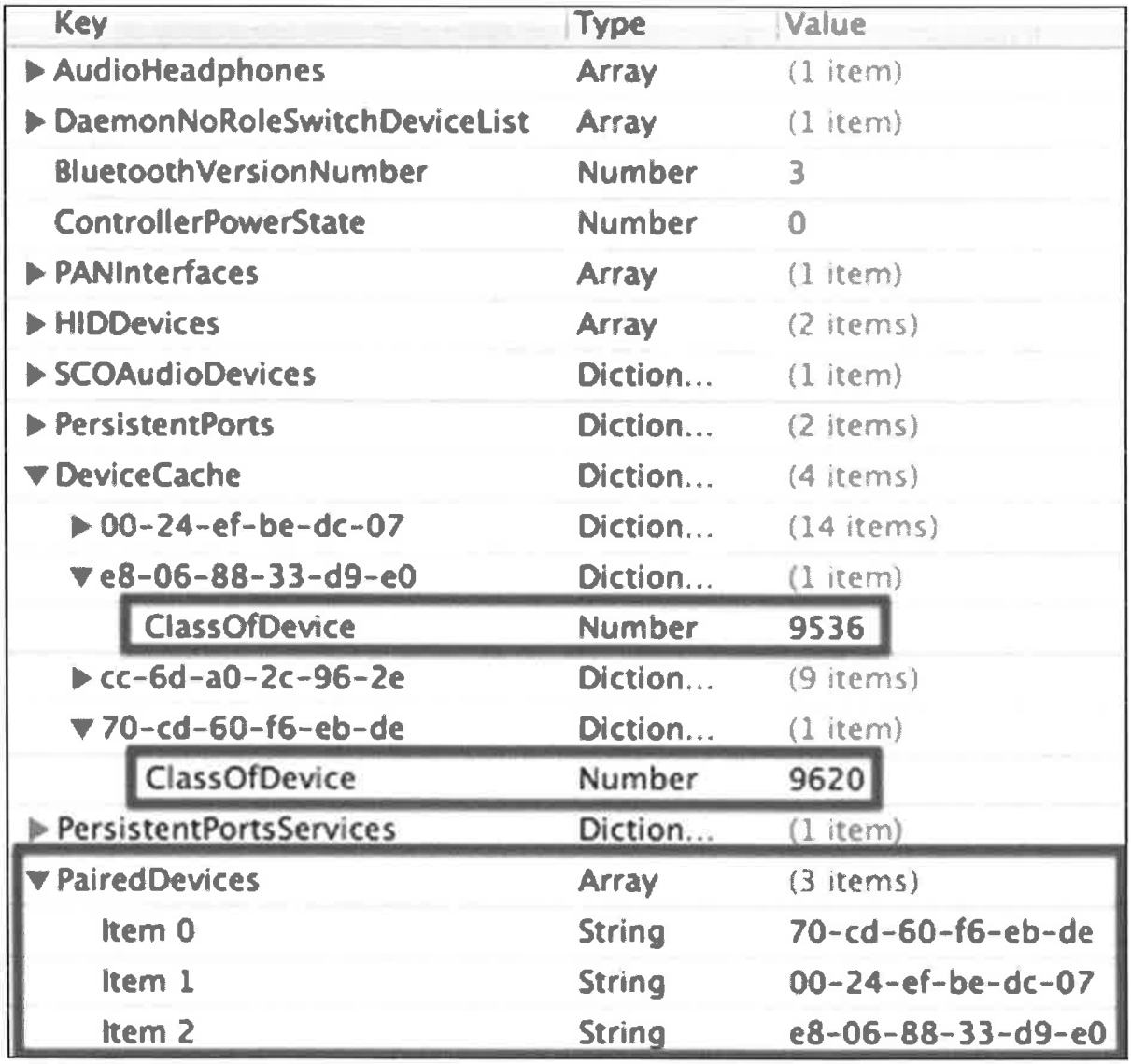
The Bluetooth preference pane. Configured Bluetooth devices may be shown in the window presented. Devices may be configured using this preferences pane.

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.

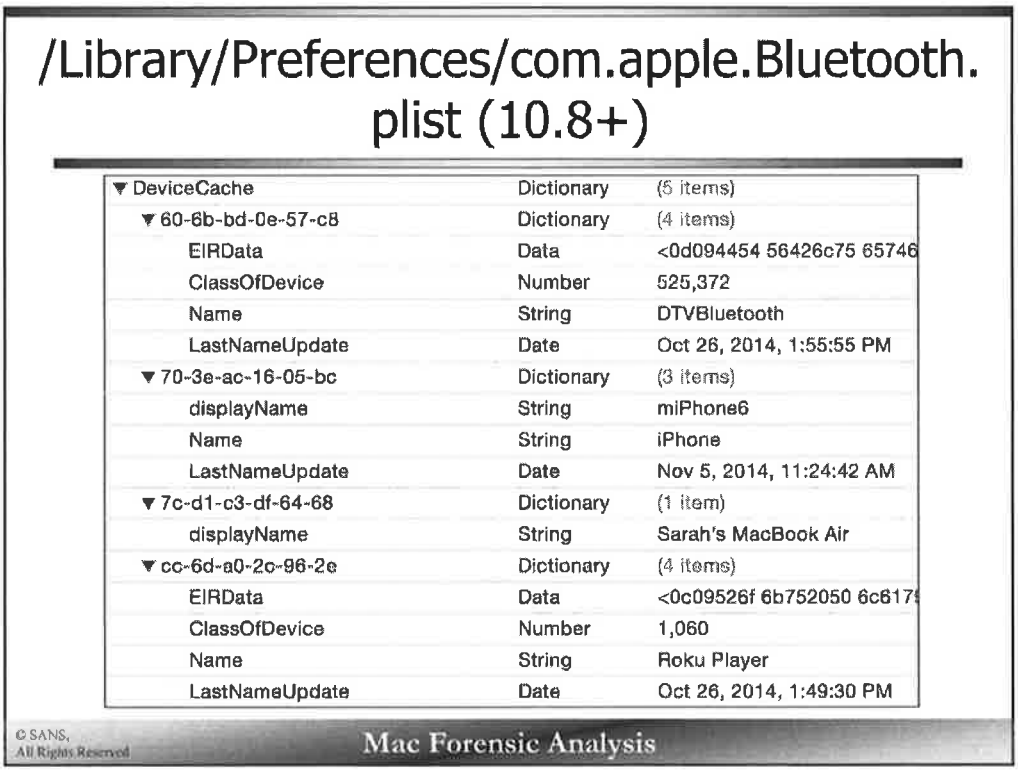
This `DeviceCache` key contains a history of the Bluetooth devices connected to the system and the `ClassOfDevice` associated with it. The device class (`ClassOfDevice` key) is the type of device that was paired, such as headphones or a mouse.

The device class can be reversed using the Bluetooth header files located in `/System/Library/Frameworks/IOBluetooth.framework/Headers/ directory.`

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

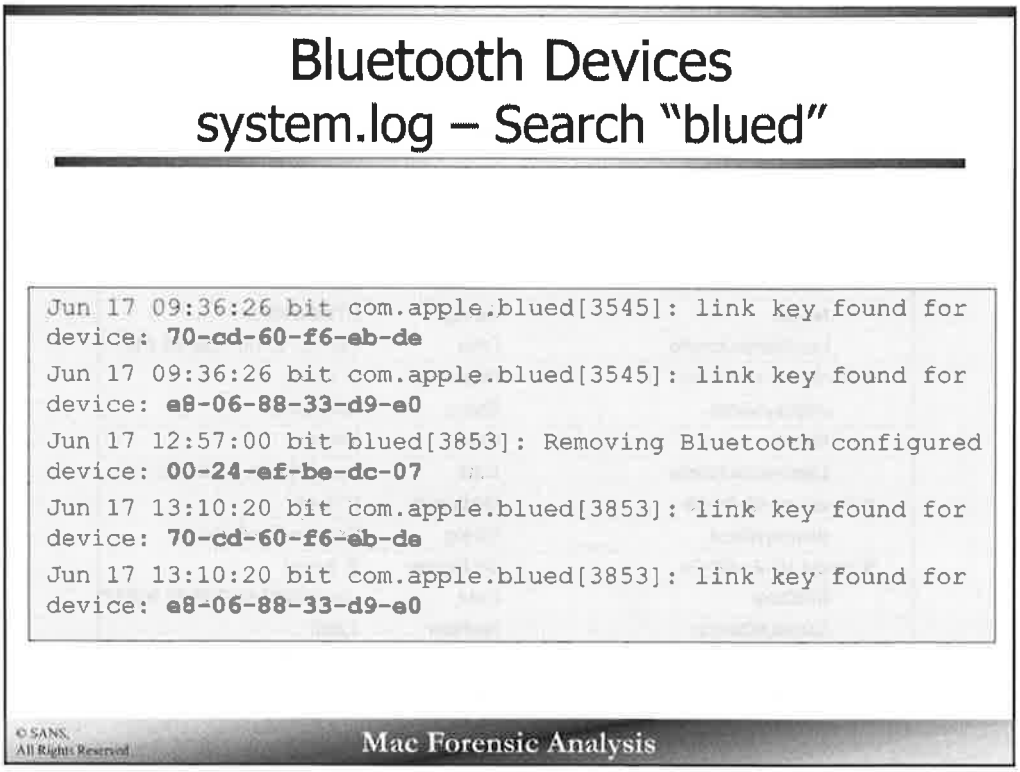


Key	Type	Value
▶ AudioHeadphones	Array	(1 item)
▶ DaemonNoRoleSwitchDeviceList	Array	(1 item)
BluetoothVersionNumber	Number	3
ControllerPowerState	Number	0
▶ PANInterfaces	Array	(1 item)
▶ HIDDevices	Array	(2 items)
▶ SCOAudioDevices	Diction...	(1 item)
▶ PersistentPorts	Diction...	(2 items)
▼ DeviceCache	Diction...	(4 items)
▶ 00-24-ef-be-dc-07	Diction...	(14 items)
▼ e8-06-88-33-d9-e0	Diction...	(1 item)
ClassOfDevice	Number	9536
▶ cc-6d-a0-2c-96-2e	Diction...	(9 items)
▼ 70-cd-60-f6-eb-de	Diction...	(1 item)
ClassOfDevice	Number	9620
▶ PersistentPortsServices	Diction...	(1 item)
▼ PairedDevices	Array	(3 items)
Item 0	String	70-cd-60-f6-eb-de
Item 1	String	00-24-ef-be-dc-07
Item 2	String	e8-06-88-33-d9-e0

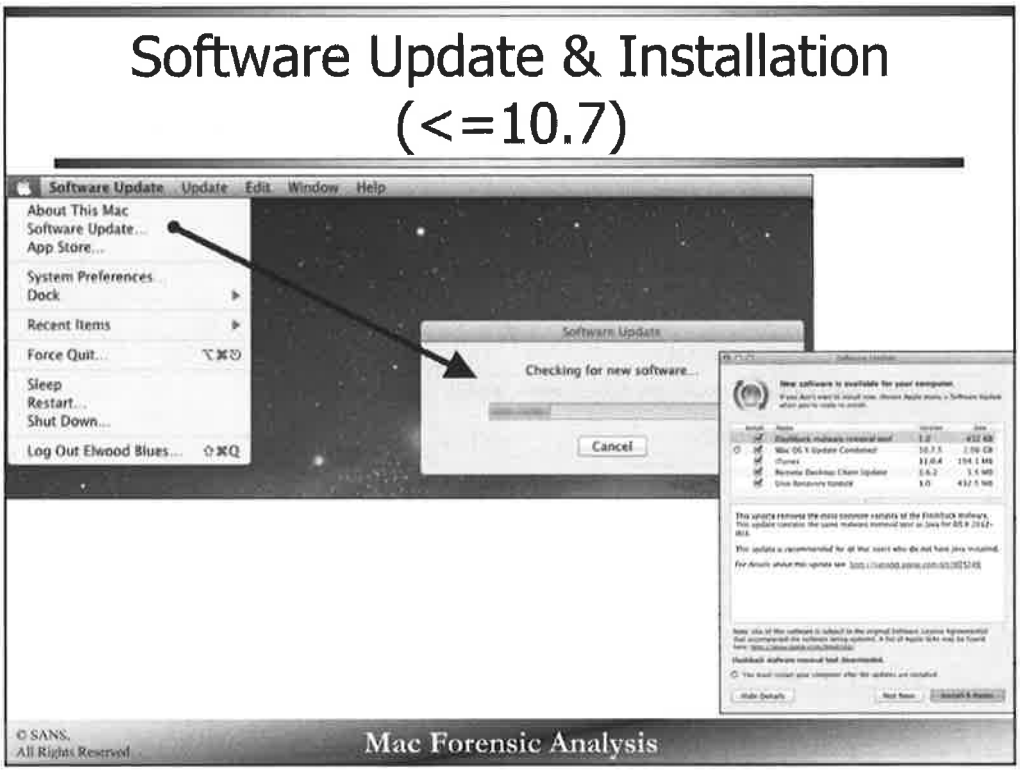


10.8 introduces more information in the `com.apple.Bluetooth.plist`. Each Bluetooth MAC address under the `DeviceCache` key may contain detailed information about the device such as the device time, and last communication timestamps.

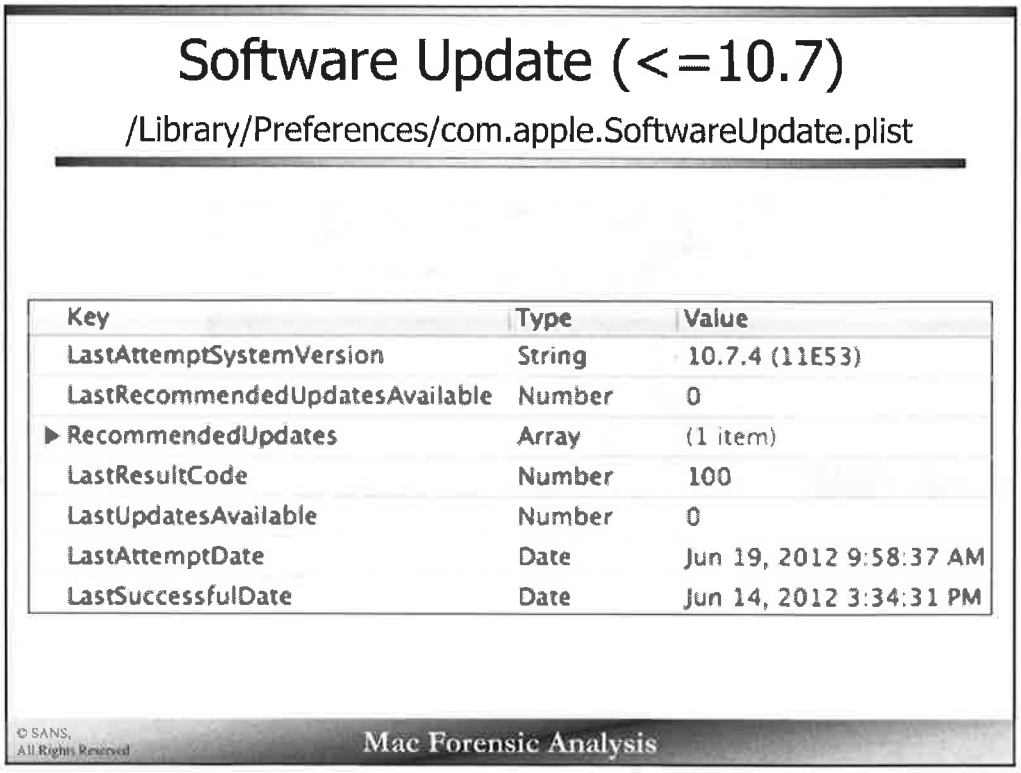
This is a good place to look for other devices a user may have had near them.



Bluetooth devices have become increasingly popular and most modern Mac devices have the technology built in. The `system.log` records the Bluetooth connected devices using the "`blued`" daemon. Searching for the "`blued`" term can show us when each devices was connected (or disconnected) and its MAC address.



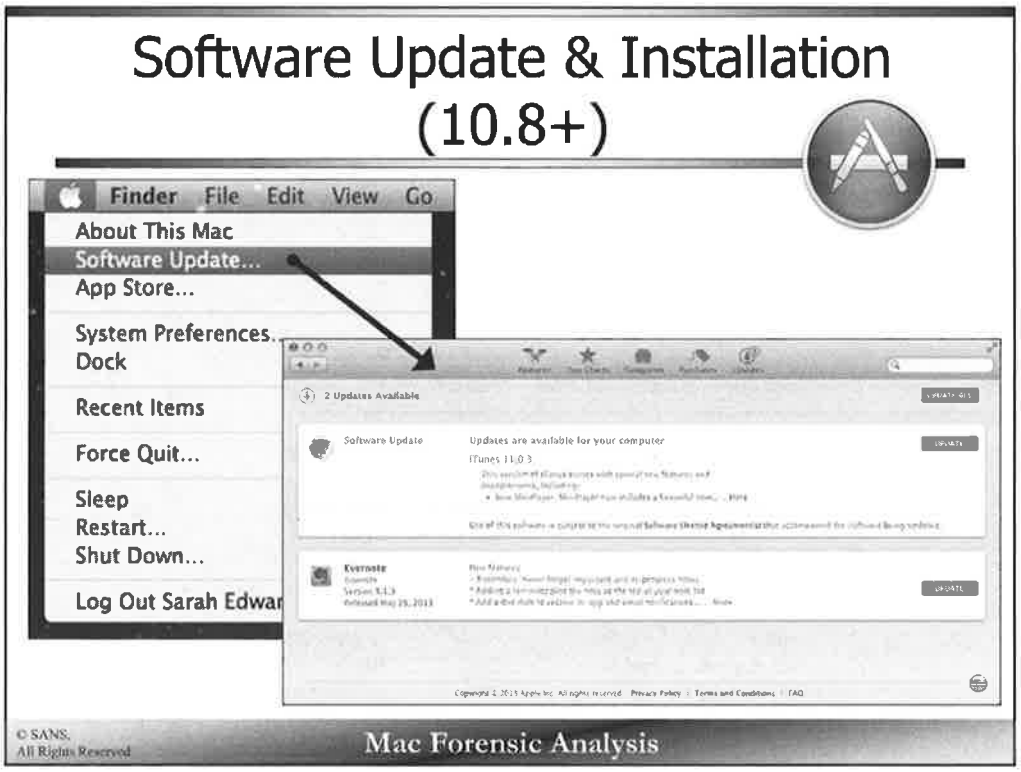
The Apple menu contains the item "`Software Update...`", when selected a software update window will appear with the message, "`checking for new software...`". If updates are found, these updates will be shown in another window where the user is able to pick and choose what to install.



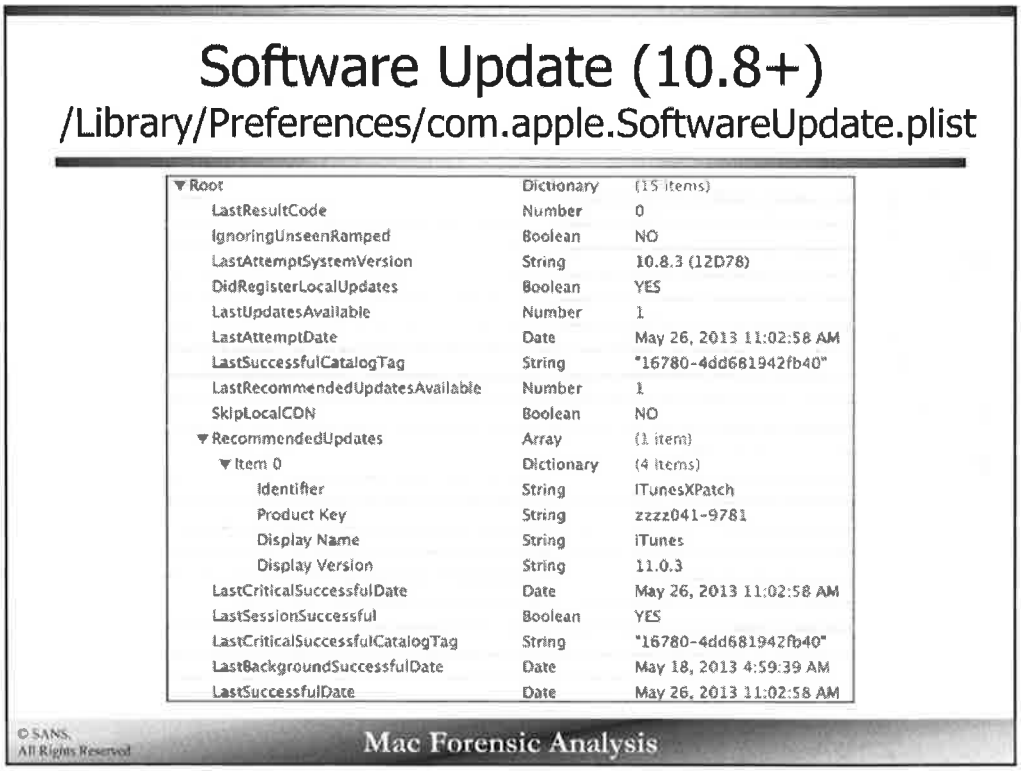
The `com.apple.SoftwareUpdate.plist` property list located in the `/Library/Preferences/` directory contains information about the software update function.

The key "`LastAttemptDate`" contains the date when the Software Update application last **attempted** to update software, while the "`LastSuccessfulDate`" contains the date when the software was **successfully** checked for an update.

The property list also shows the last system version that was used when it last attempted an update. It also shows the number of updates available for installation.



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, both the Apple system updates and any application that had been downloaded and installed via the Mac App Store.

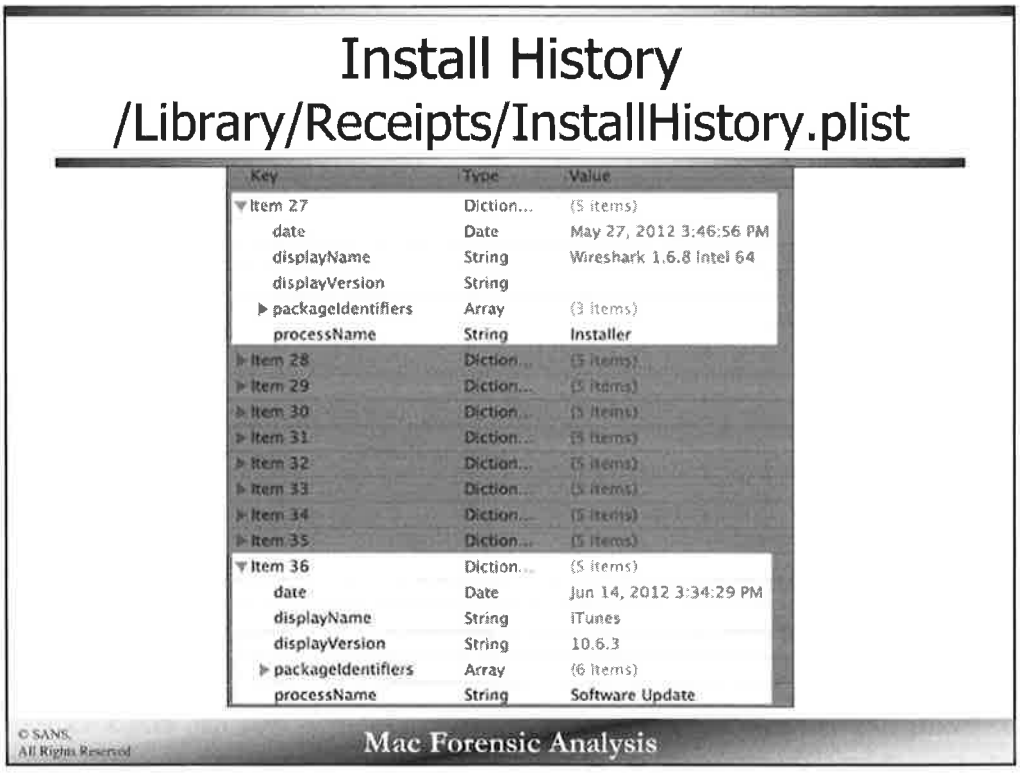


The `com.appleSoftwareUpdate.plist` property list for 10.8+ systems contains additional data.

This property list contains data such as when critical updates and updates downloaded in the background were last installed.

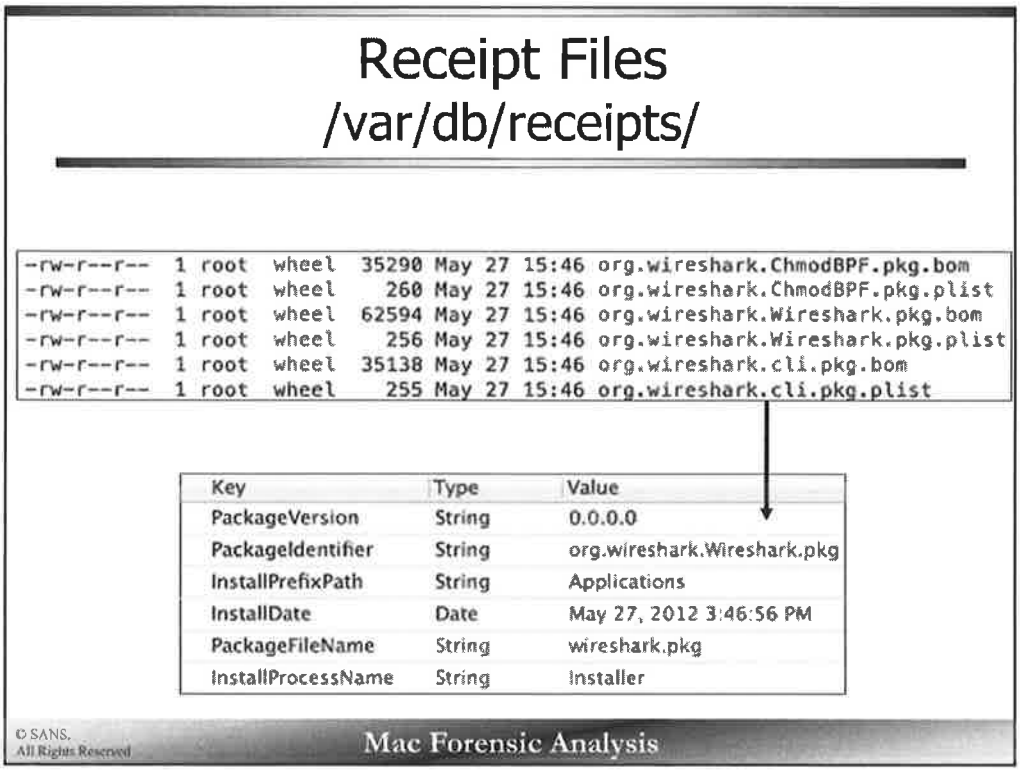
The key `LastBackgroundSuccessfulDate` contains the timestamp of the last time updates were installed in the background.

▼ Root	Dictionary	(15 items)
LastResultCode	Number	0
IgnoringUnseenRamped	Boolean	NO
LastAttemptSystemVersion	String	10.8.3 (12D78)
DidRegisterLocalUpdates	Boolean	YES
LastUpdatesAvailable	Number	1
LastAttemptDate	Date	May 26, 2013 11:02:58 AM
LastSuccessfulCatalogTag	String	"16780-4dd681942fb40"
LastRecommendedUpdatesAvailable	Number	1
SkipLocalCDN	Boolean	NO
▼ RecommendedUpdates	Array	(1 item)
▼ Item 0	Dictionary	(4 items)
Identifier	String	iTunesXPatch
Product Key	String	zzzz041-9781
Display Name	String	iTunes
Display Version	String	11.0.3
LastCriticalSuccessfulDate	Date	May 26, 2013 11:02:58 AM
LastSessionSuccessful	Boolean	YES
LastCriticalSuccessfulCatalogTag	String	"16780-4dd681942fb40"
LastBackgroundSuccessfulDate	Date	May 18, 2013 4:59:39 AM
LastSuccessfulDate	Date	May 26, 2013 11:02:58 AM



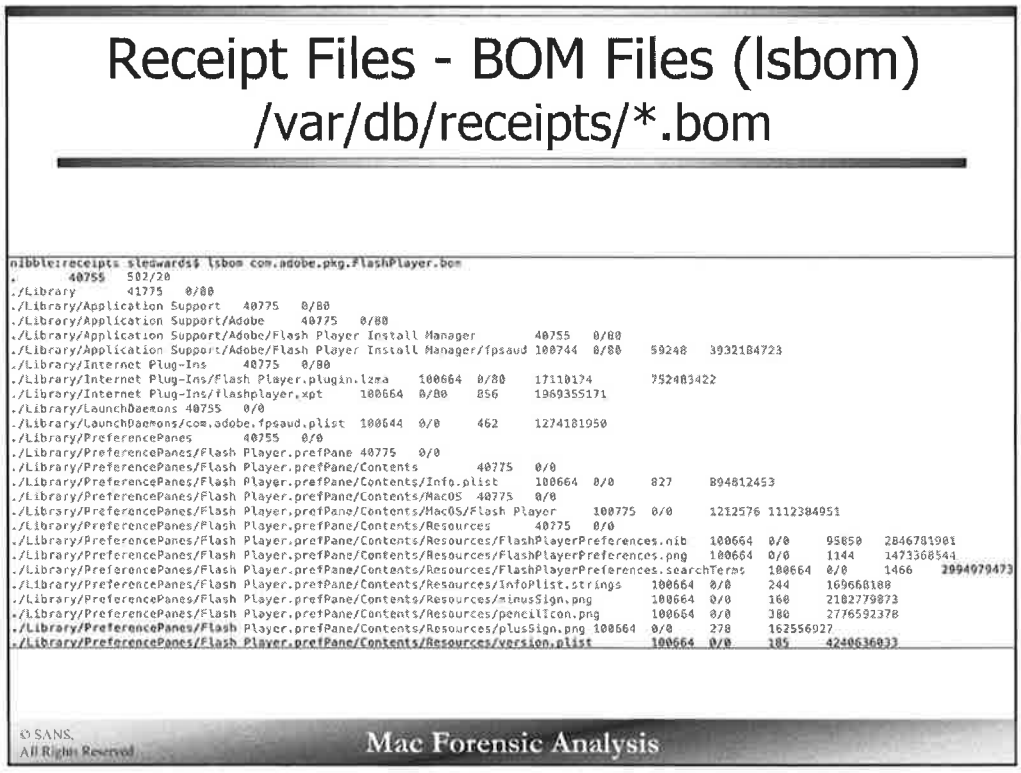
The `InstallHistory.plist` property list located in `/Library/Receipts/` contains a software install history that includes the timestamp, software package name, and what process was used to install the software.

The process name, "`Installer`" is normally seen when a user manually installs a piece of software such as Wireshark while the process "`Software Update`" is seen when the system installs its updates.



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.



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

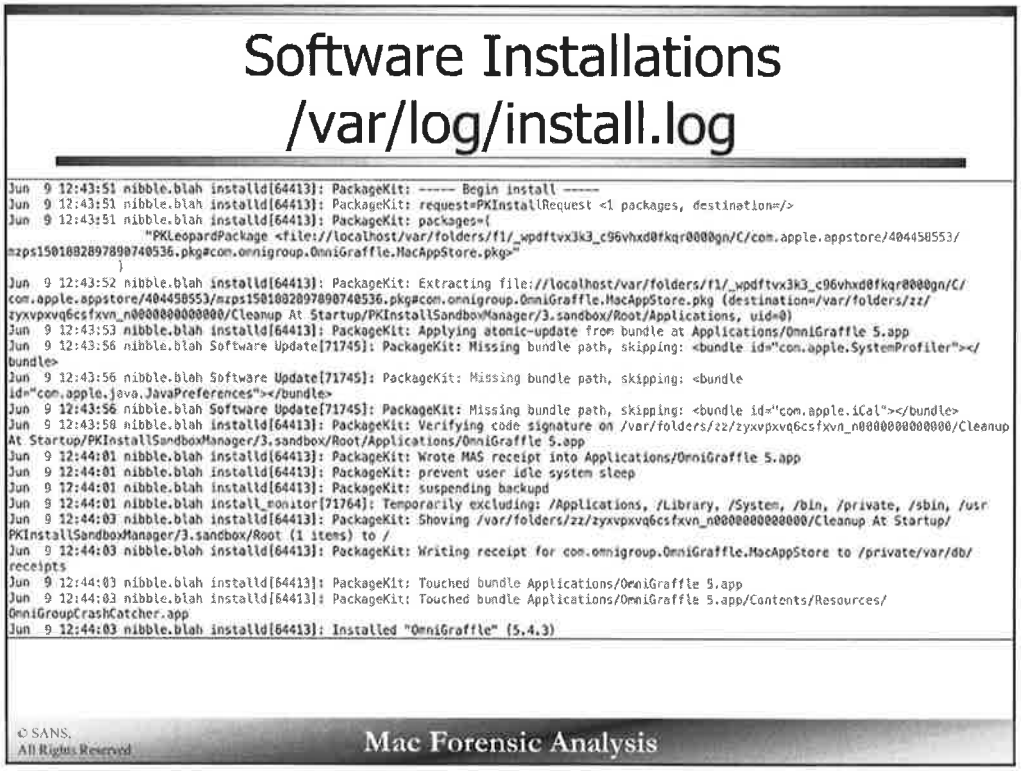
Each type of file will contain different information in the BoM file.

- 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.

References:

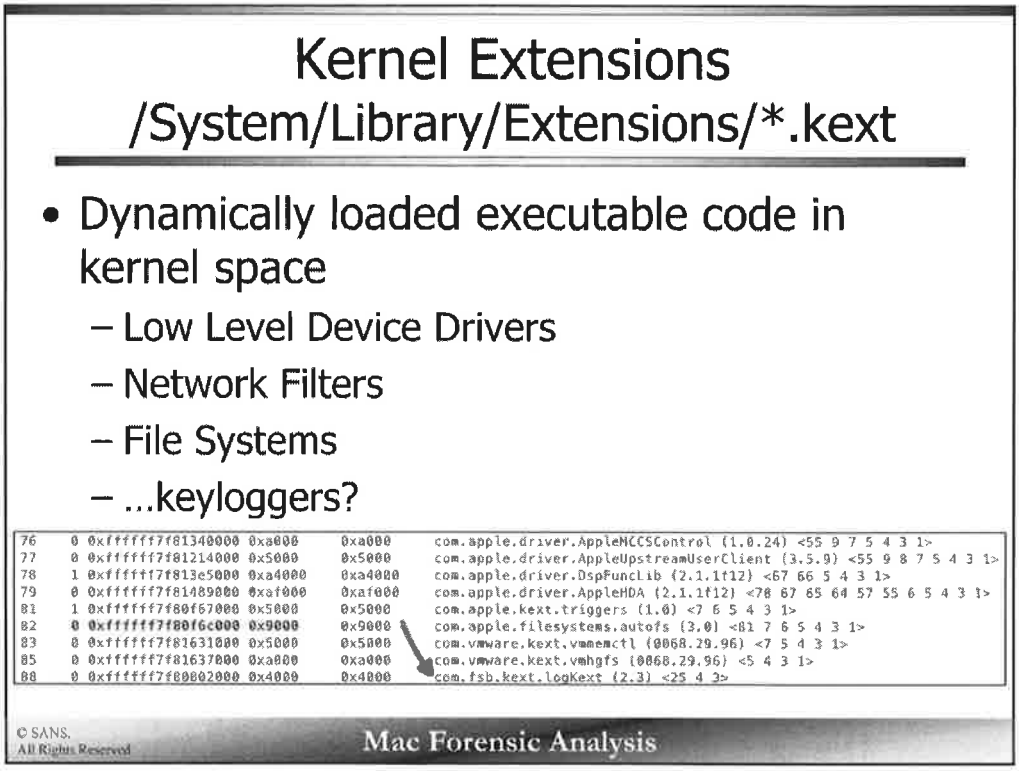
Man Page for lsbom

```
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 `install.log` located in `/var/log/` contains the packages and applications installed on the system, whether administrator credentials were needed, and the path to where the application was located when it was installed.

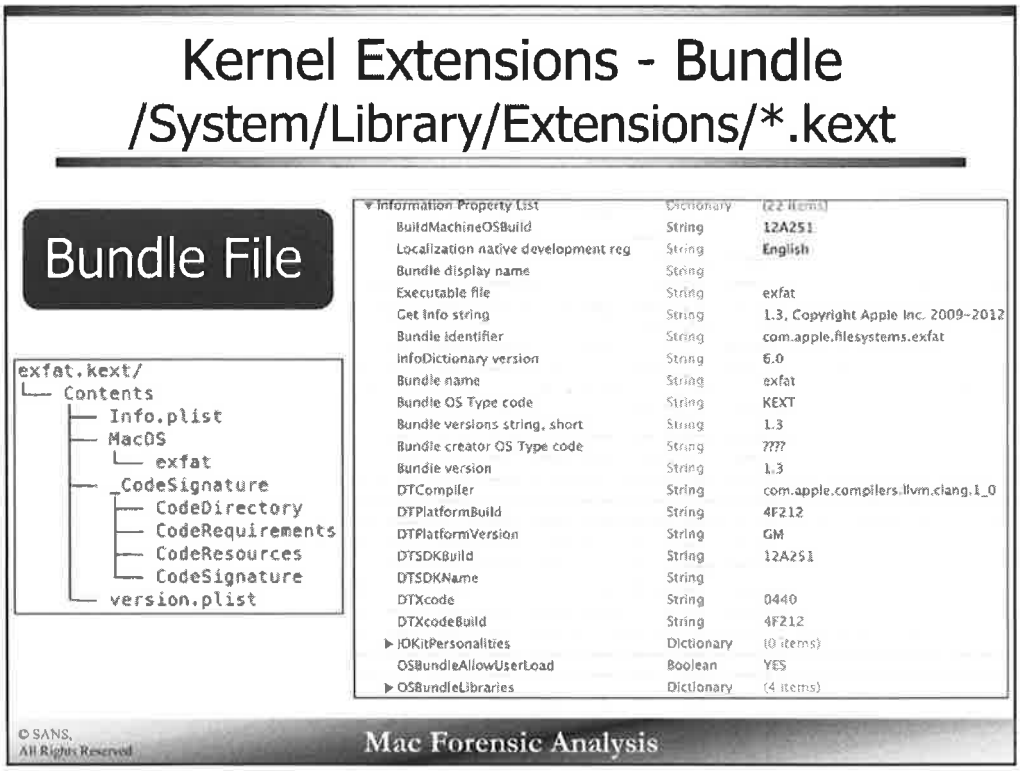
```
Jun  9 12:43:51 nibble.blah installd[64413]: PackageKit: ----- Begin install -----
Jun  9 12:43:51 nibble.blah installd[64413]: PackageKit: request=PKInstallRequest <1 packages, destination=/>
Jun  9 12:43:51 nibble.blah installd[64413]: PackageKit: packages=(
                    "PKLeopardPackage <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)
```



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.

```
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>
```



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

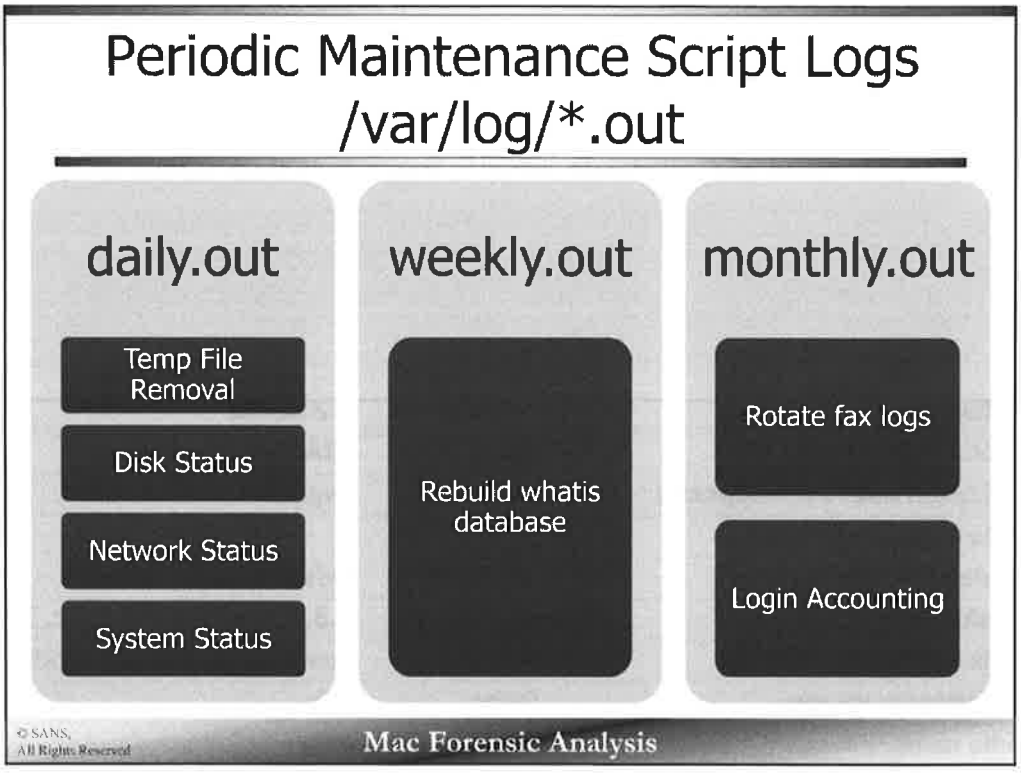
The screenshot on the left shows the file structure of the `exfat` file system kernel extension. Each will contain a Contents directory with an `Info.plist` file as shown on the right. The `Info.plist` file contains the identifying information of the kernel extension including the version number, executable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

References:

Apple Developer Documentation – Kernel Programming

https://developer.apple.com/library/mac/documentation/Darwin/Conceptual/KernelProgramming/KernelProgramming.pdf

▼ 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)



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

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

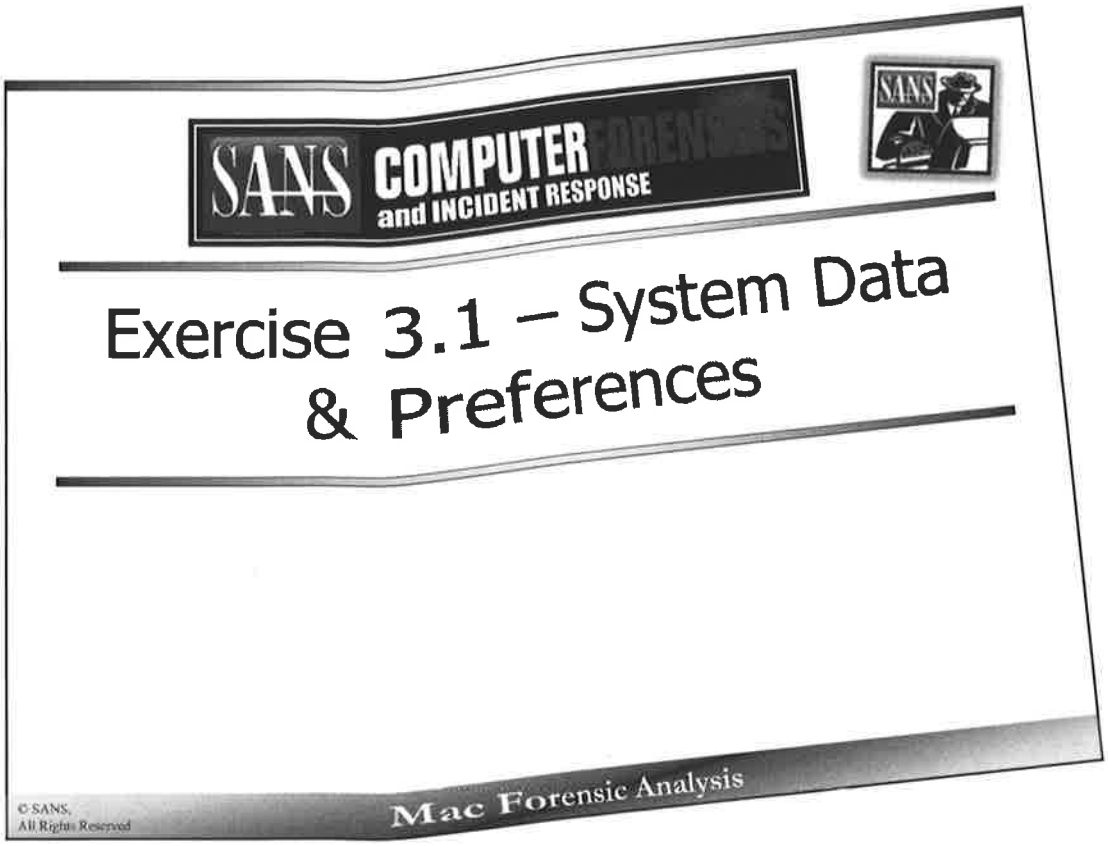
- **`daily.out`** – Contains output from temporary file removal, disk, network, and system status.
- **`weekly.out`** – Contains output from rebuilding the `whatis` database. This database contains a description of system commands.
- **`monthly.out`** – Contains output from rotating the fax logs and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

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

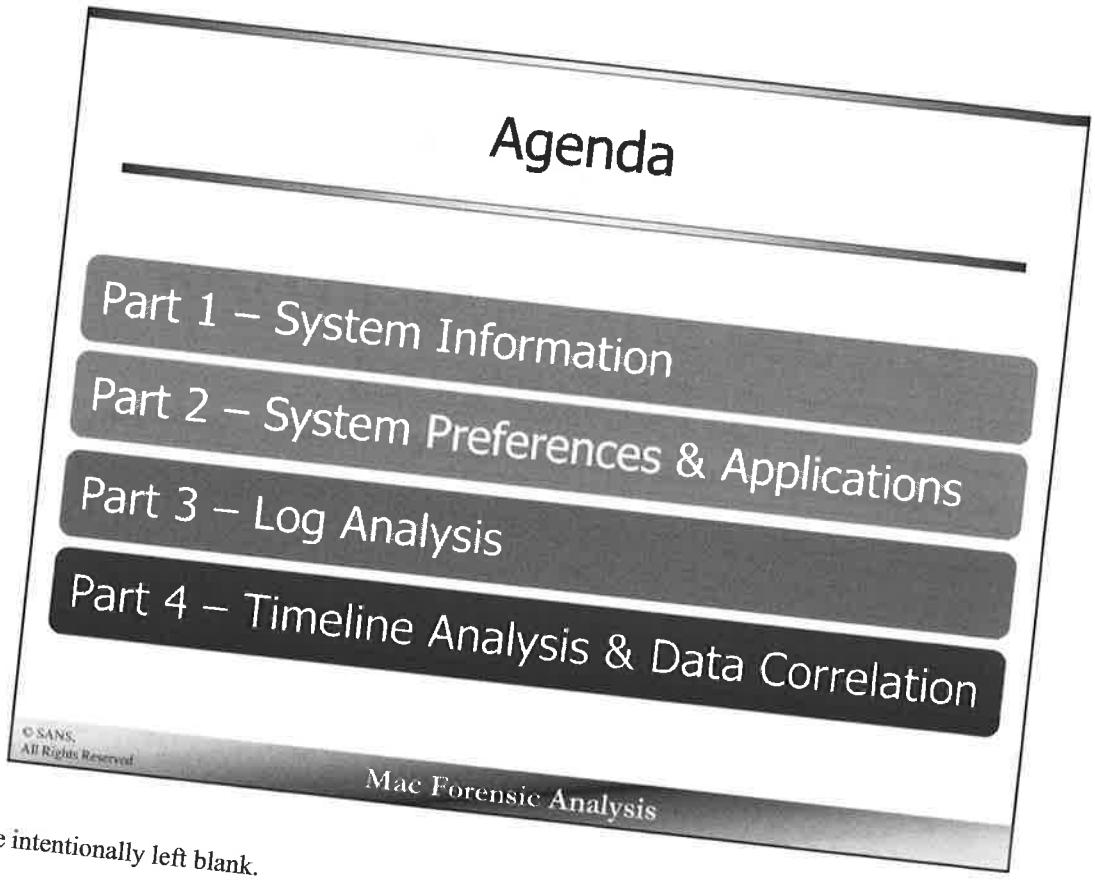
Reference:

periodic Man Page

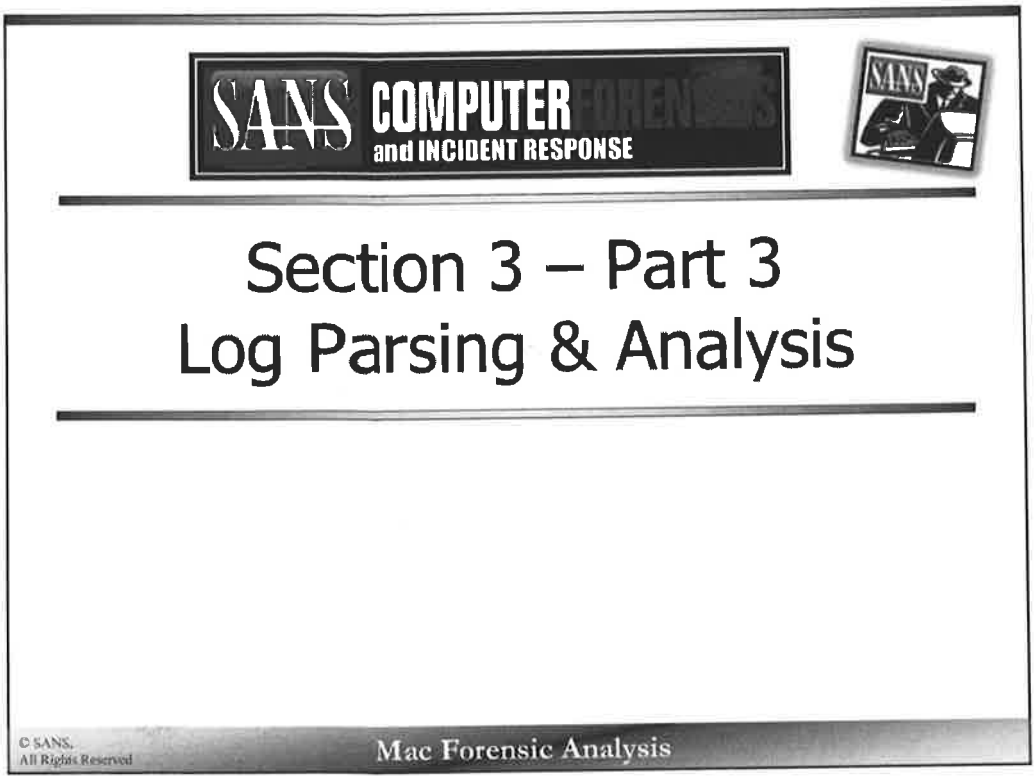
periodic.conf Man Page



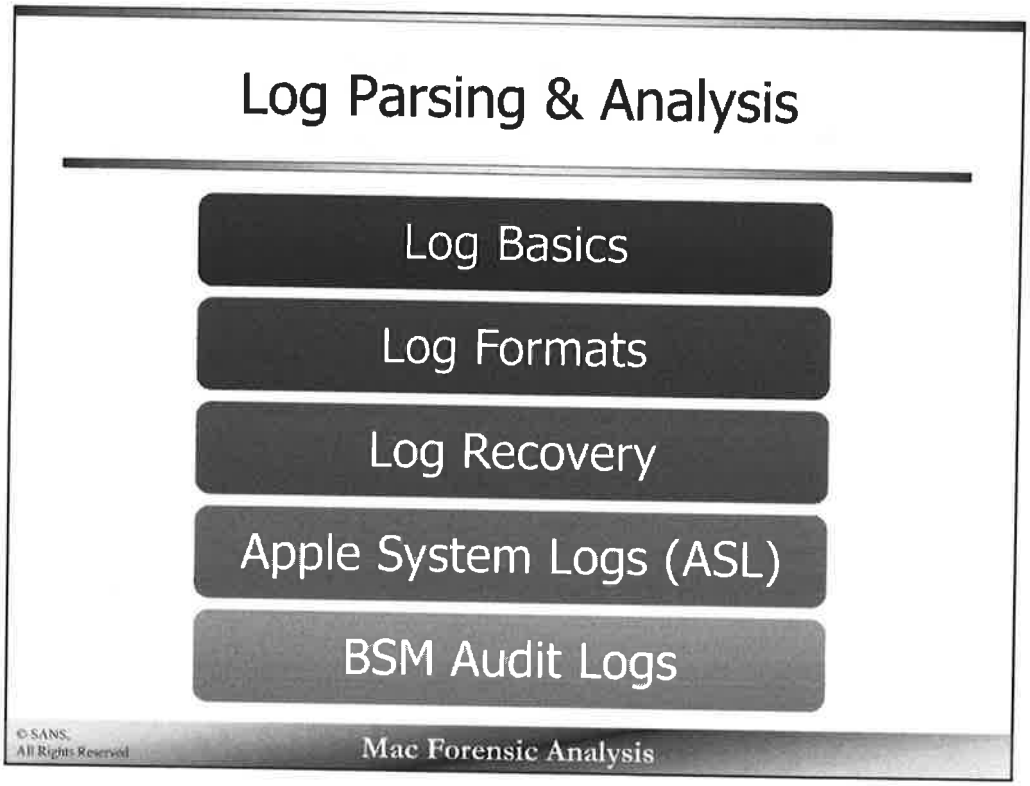
This page intentionally left blank.



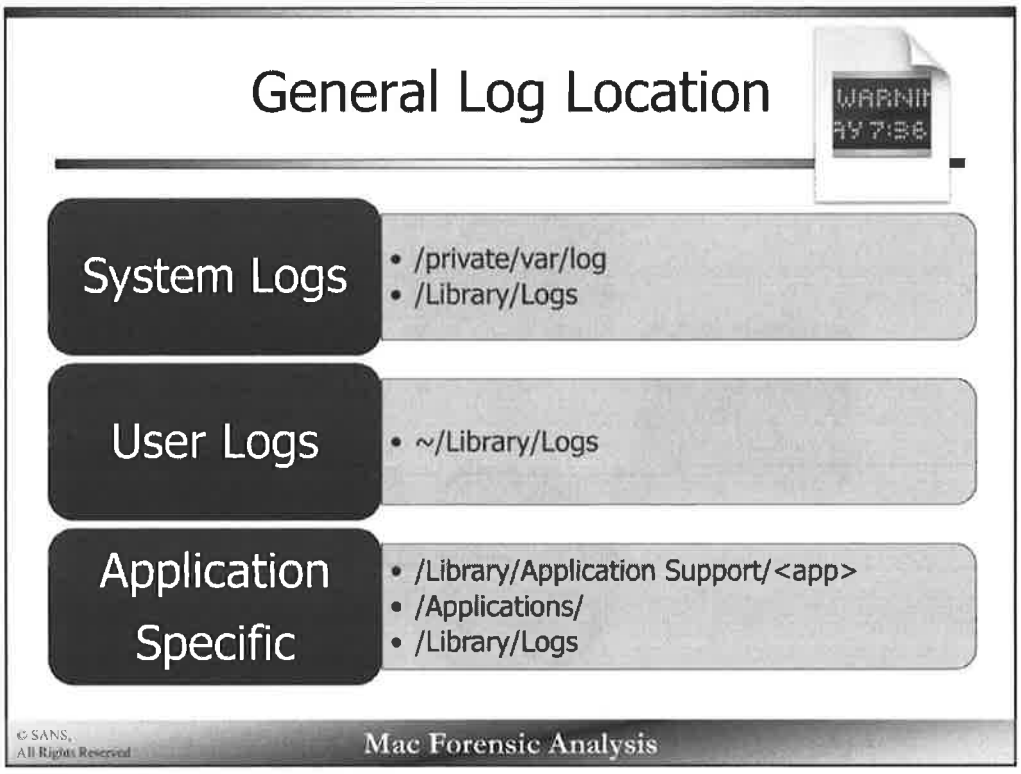
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.

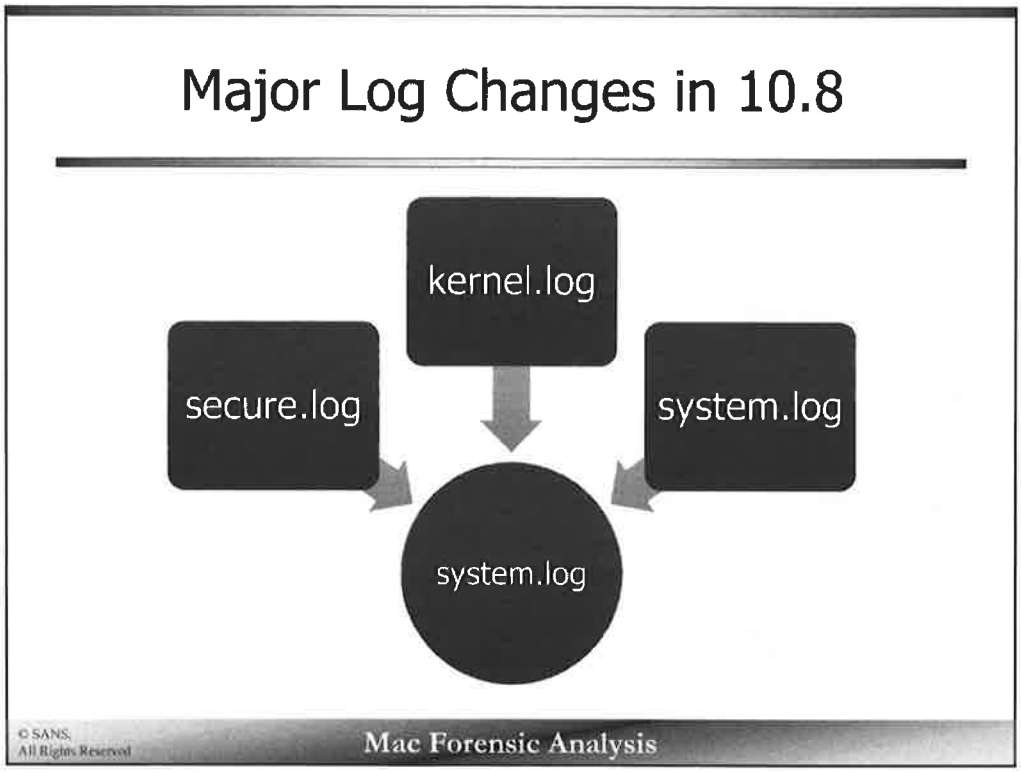


There are three primary locations in OS X where logs are found.

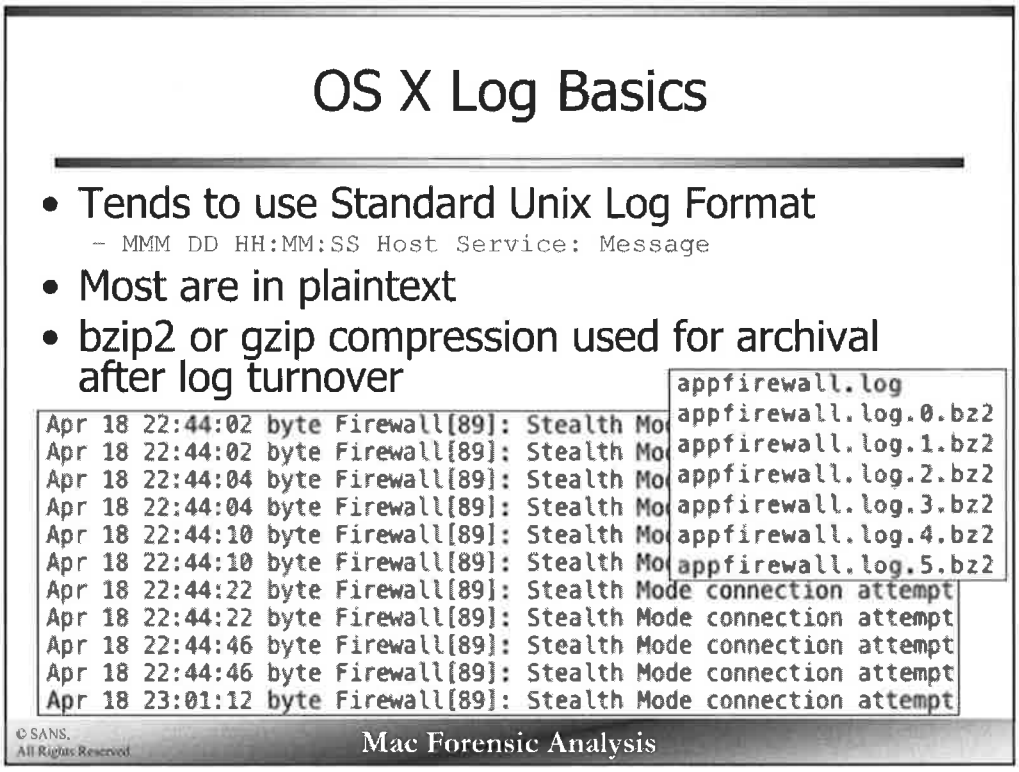
System logs, those that have to do with the operating system, can be found in `/private/var/log` and `/Library/Logs`.

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.



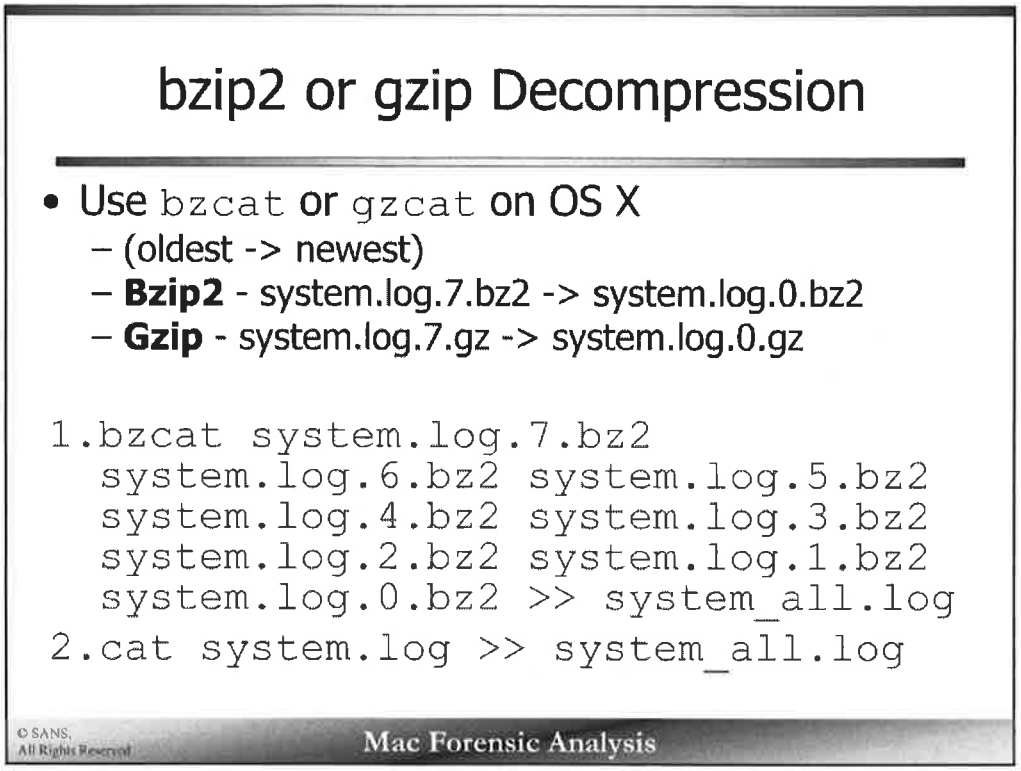
In 10.8 there no longer exists separate log files for `secure.log`, `kernel.log`, and `system.log`. These files have been combined into one, `system.log` file.



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`).

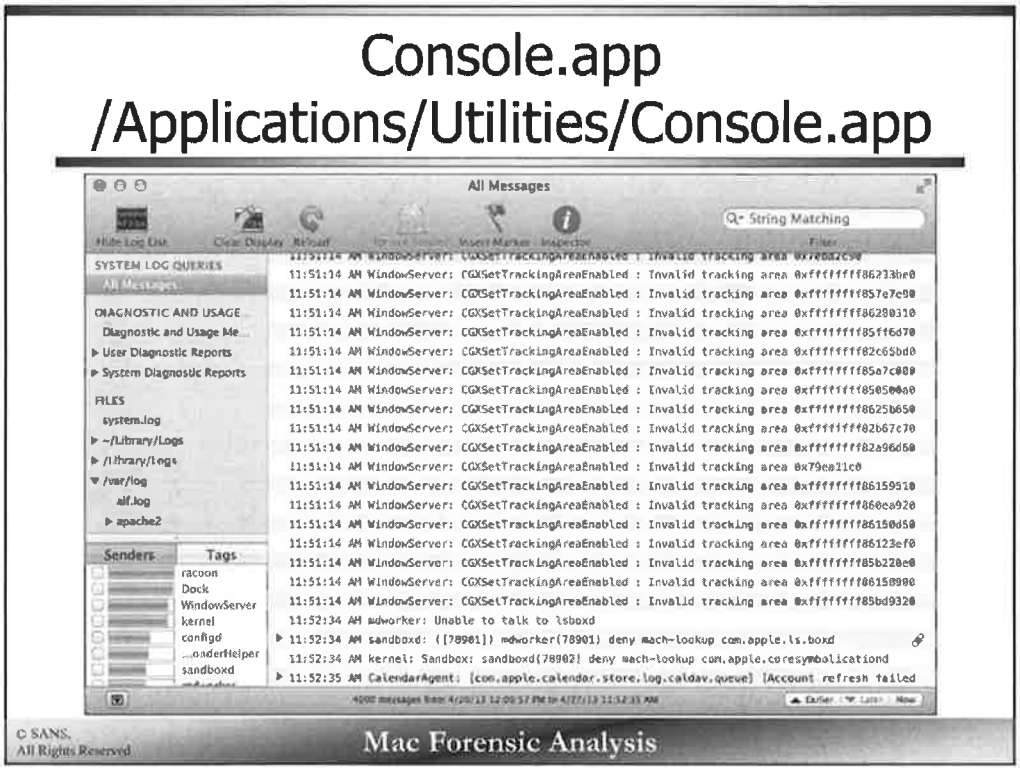
Generally speaking, on 10.8 and prior systems, bzip2 compression is used to archive its log data while on 10.9 systems, gzip is used.



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 (ie: `system.log.7.bz2`) the older the log archive.

- 0 = newest
- 7 = oldest

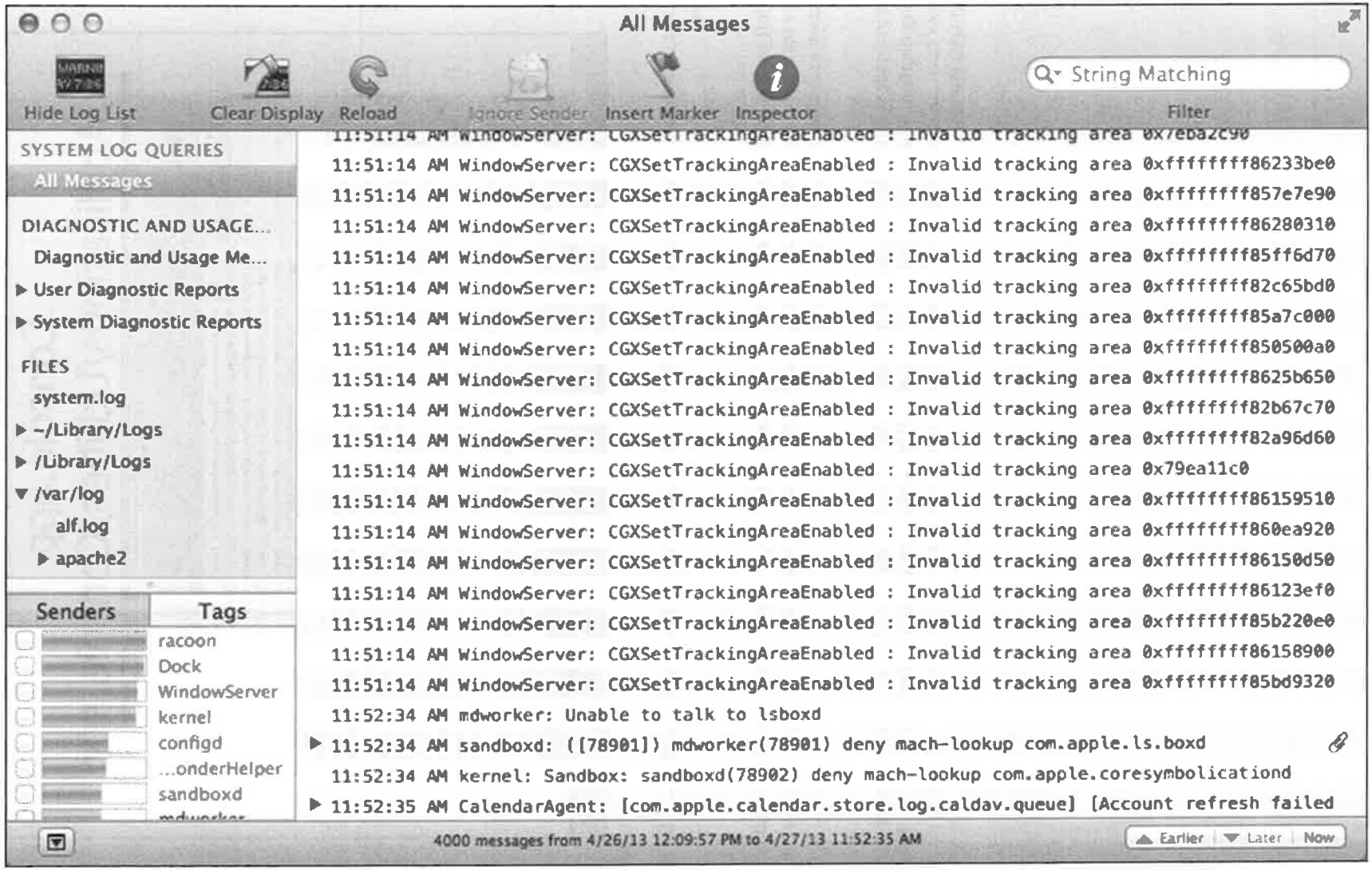
The first command concatenates the contents of the archive files from oldest to newest into a file named `system_all.log`, the second command concatenates the current log file `system.log` to the `system_all.log` file to create a complete `system.log` file for the system.



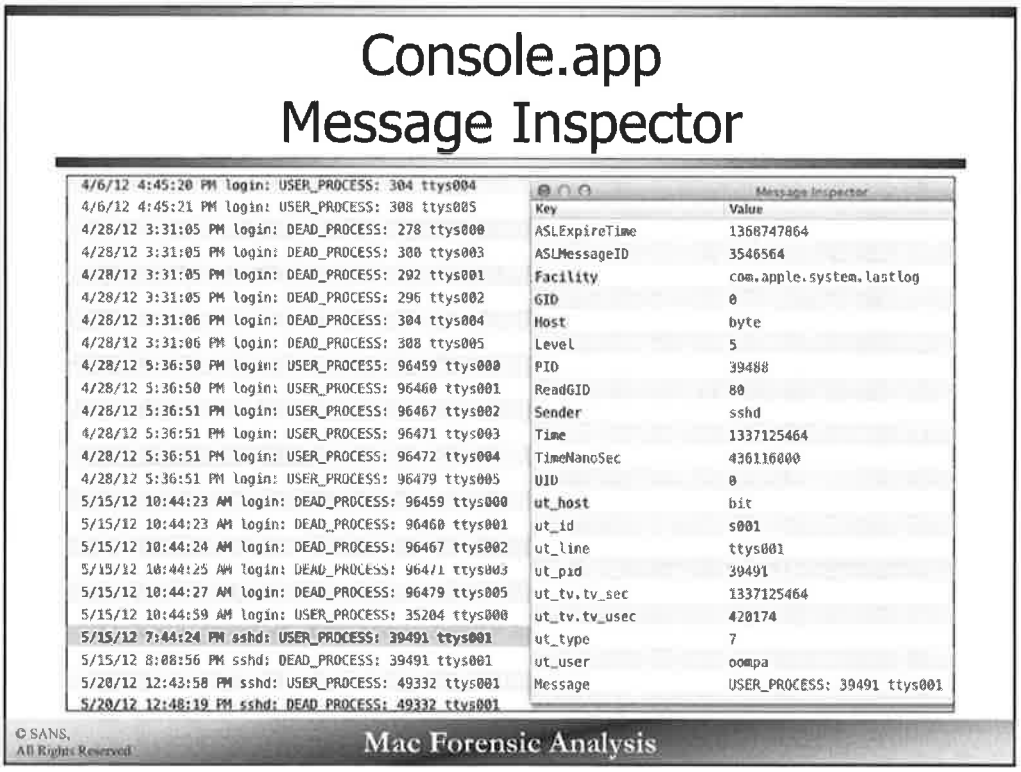
The `Console.app` is a native log viewer on OS X. It can be an incredibly powerful and useful tool for log analysis.

In the left pane is an auto-populated log list that includes system, user, and application logs. Below that is a window that shows the most popular message Sender and Tags when used with the "`All Messages`" display. The button in the lower-left can access this pane.

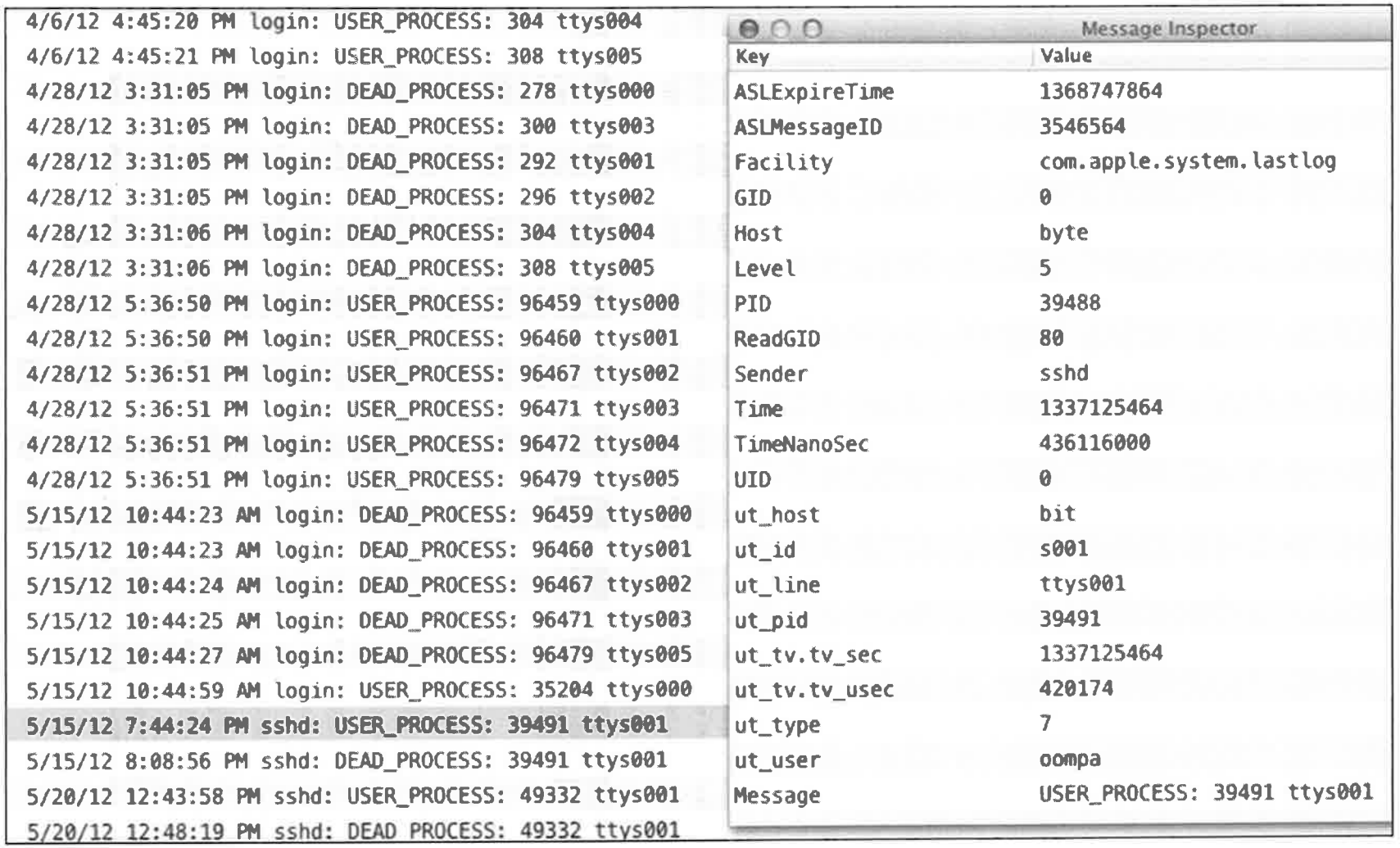
The main window shows the logs, one line per entry. The "`All Messages`" display is more dynamic than other log displays. This "`All Messages`" includes all the syslog entries that the user has access to. These messages have additional metadata that can be viewed using the `Message Inspector` – the blue circle icon marked with the letter "`i`".

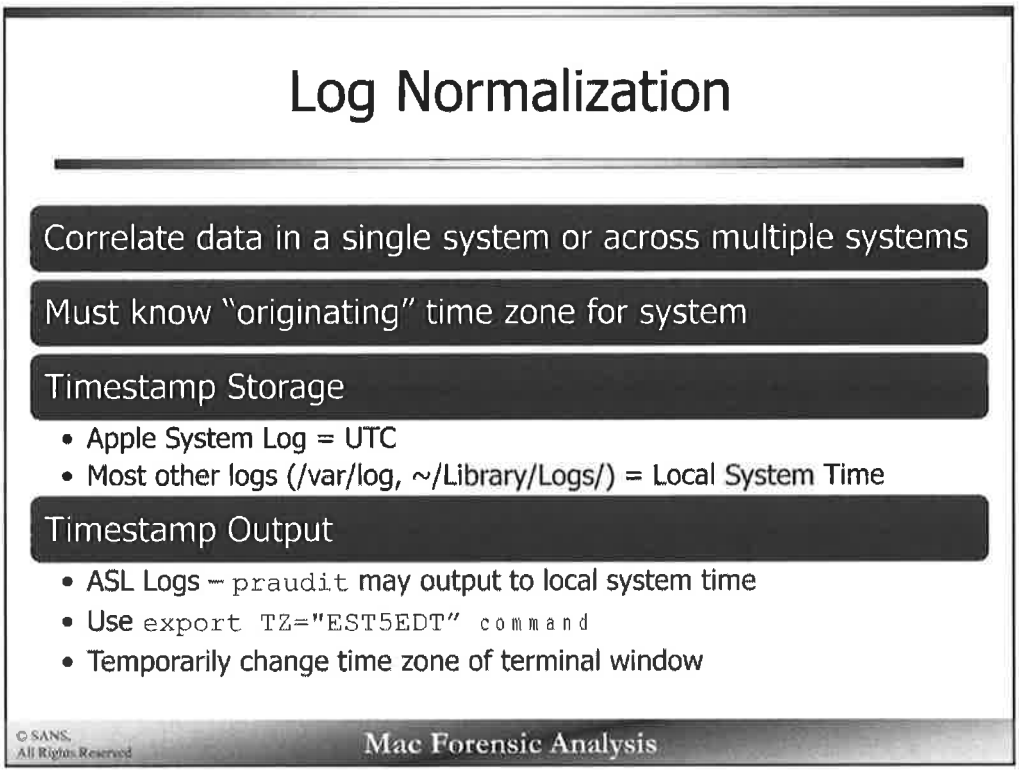

All Messages
Hide Log List
Clear Display
Reload
Ignore Sender
Insert Marker
Inspector
String Matching
Filter
SYSTEM LOG QUERIES
All Messages
DIAGNOSTIC AND USAGE...
Diagnostic and Usage Me...
User Diagnostic Reports
System Diagnostic Reports
FILES
system.log
~/Library/Logs
/Library/Logs
/var/log
alf.log
apache2
Senders
Tags
racoon
Dock
WindowServer
kernel
configd
...onderHelper
sandboxd
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff86233be0
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff857e7e90
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff86280310
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff85ff6d70
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff82c65bd0
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff85a7c000
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff850500a0
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff8625b650
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff82b67c70
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff82a96d60
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0x79ea11c0
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff86159510
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff860ea920
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff86150d50
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff86123ef0
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff85b220e0
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff86158900
11:51:14 AM WindowServer: CGXSetTrackingAreaEnabled : Invalid tracking area 0xffffffff85bd9320
11:52:34 AM mdworker: Unable to talk to lsboxd
11:52:34 AM sandboxd: ([78901]) mdworker(78901) deny mach-lookup com.apple.ls.boxd
11:52:34 AM kernel: Sandbox: sandboxd(78902) deny mach-lookup com.apple.coresymbolicationd
11:52:35 AM CalendarAgent: [com.apple.calendar.store.log.caldav.queue] [Account refresh failed
4000 messages from 4/26/13 12:09:57 PM to 4/27/13 11:52:35 AM
Earlier
Later
Now

110



This view shows the "`All Messages`", syslog entries alongside the `Message Inspector` window. This window shows the additional metadata included with each syslog message. This metadata can be extracted from the raw data as shown in a later slide.


4/6/12 4:45:20 PM login: USER_PROCESS: 304 ttys004
4/6/12 4:45:21 PM login: USER_PROCESS: 308 ttys005
4/28/12 3:31:05 PM login: DEAD_PROCESS: 278 ttys000
4/28/12 3:31:05 PM login: DEAD_PROCESS: 300 ttys003
4/28/12 3:31:05 PM login: DEAD_PROCESS: 292 ttys001
4/28/12 3:31:05 PM login: DEAD_PROCESS: 296 ttys002
4/28/12 3:31:06 PM login: DEAD_PROCESS: 304 ttys004
4/28/12 3:31:06 PM login: DEAD_PROCESS: 308 ttys005
4/28/12 5:36:50 PM login: USER_PROCESS: 96459 ttys000
4/28/12 5:36:50 PM login: USER_PROCESS: 96460 ttys001
4/28/12 5:36:51 PM login: USER_PROCESS: 96467 ttys002
4/28/12 5:36:51 PM login: USER_PROCESS: 96471 ttys003
4/28/12 5:36:51 PM login: USER_PROCESS: 96472 ttys004
4/28/12 5:36:51 PM login: USER_PROCESS: 96479 ttys005
5/15/12 10:44:23 AM login: DEAD_PROCESS: 96459 ttys000
5/15/12 10:44:23 AM login: DEAD_PROCESS: 96460 ttys001
5/15/12 10:44:24 AM login: DEAD_PROCESS: 96467 ttys002
5/15/12 10:44:25 AM login: DEAD_PROCESS: 96471 ttys003
5/15/12 10:44:27 AM login: DEAD_PROCESS: 96479 ttys005
5/15/12 10:44:59 AM login: USER_PROCESS: 35204 ttys000
5/15/12 7:44:24 PM sshd: USER_PROCESS: 39491 ttys001
5/15/12 8:08:56 PM sshd: DEAD_PROCESS: 39491 ttys001
5/20/12 12:43:58 PM sshd: USER_PROCESS: 49332 ttys001
5/20/12 12:48:19 PM sshd: DEAD_PROCESS: 49332 ttys001
Message Inspector
Key
Value
ASLExpireTime
1368747864
ASLMessageID
3546564
Facility
com.apple.system.lastlog
GID
0
Host
byte
Level
5
PID
39488
ReadGID
80
Sender
sshd
Time
1337125464
TimeNanoSec
436116000
UID
0
ut_host
bit
ut_id
s001
ut_line
ttys001
ut_pid
39491
ut_tv.tv_sec
1337125464
ut_tv.tv_usec
420174
ut_type
7
ut_user
oompa
Message
USER_PROCESS: 39491 ttys001

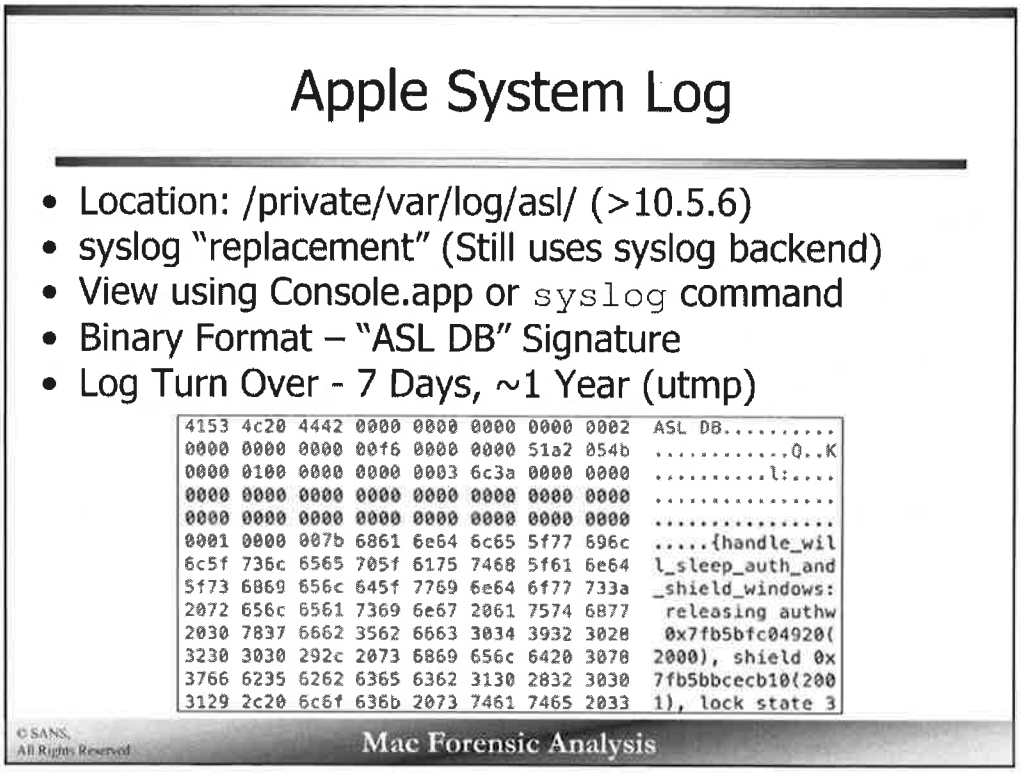


Analysts often have to correlate information across multiple systems across different time zones. To combine and understand what each system was doing at the same time, the analyst will need to normalize the log.

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 command `export TZ="EST5EDT"` 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.



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.

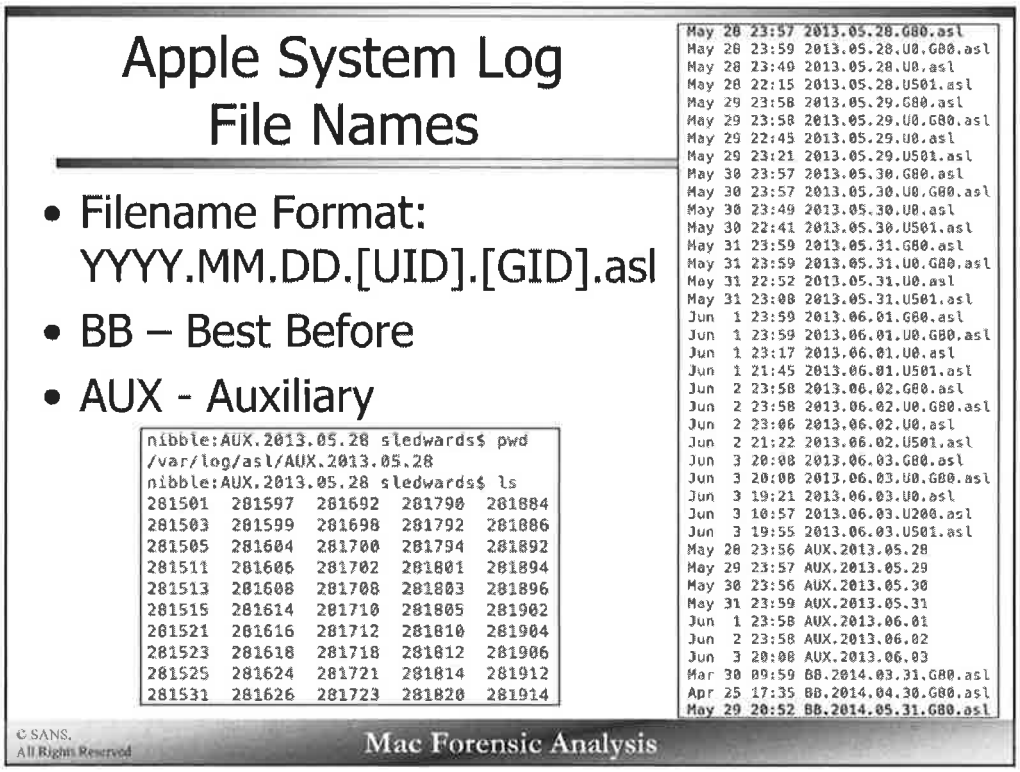
The screenshot shows the signature for an ASL log, "`ASL DB`" which will be found in the first six bytes of each log file.

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 via man asl.conf).

References:

asl.conf Man Page

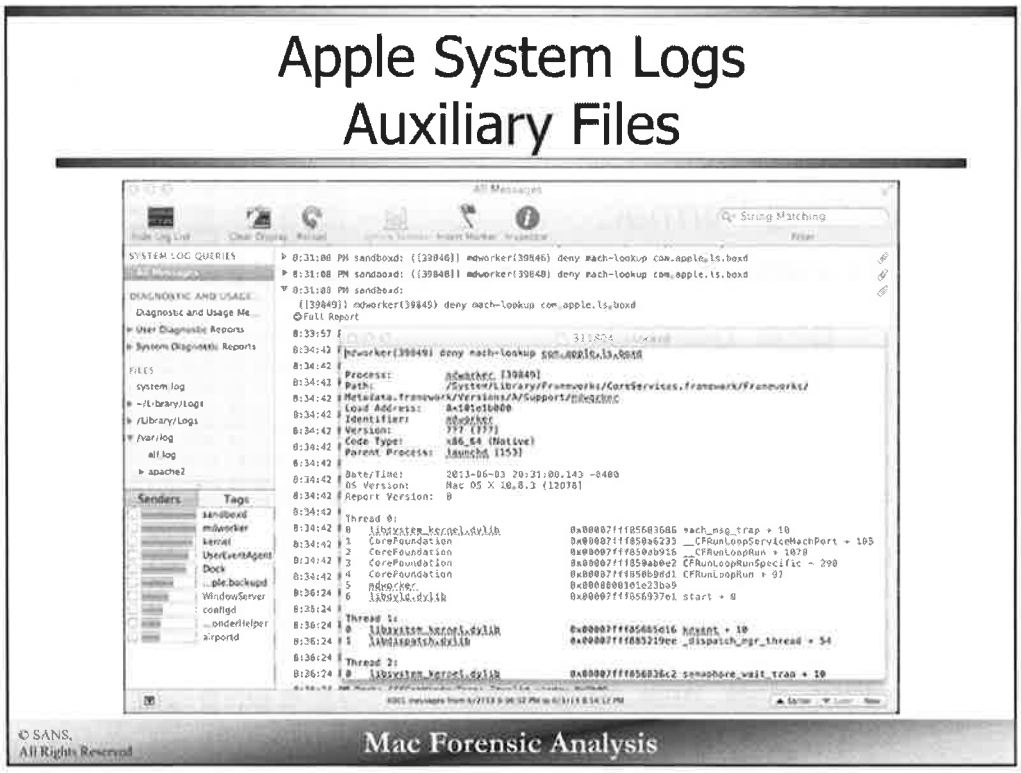
https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man5/asl.conf.5.html



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 groups 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".

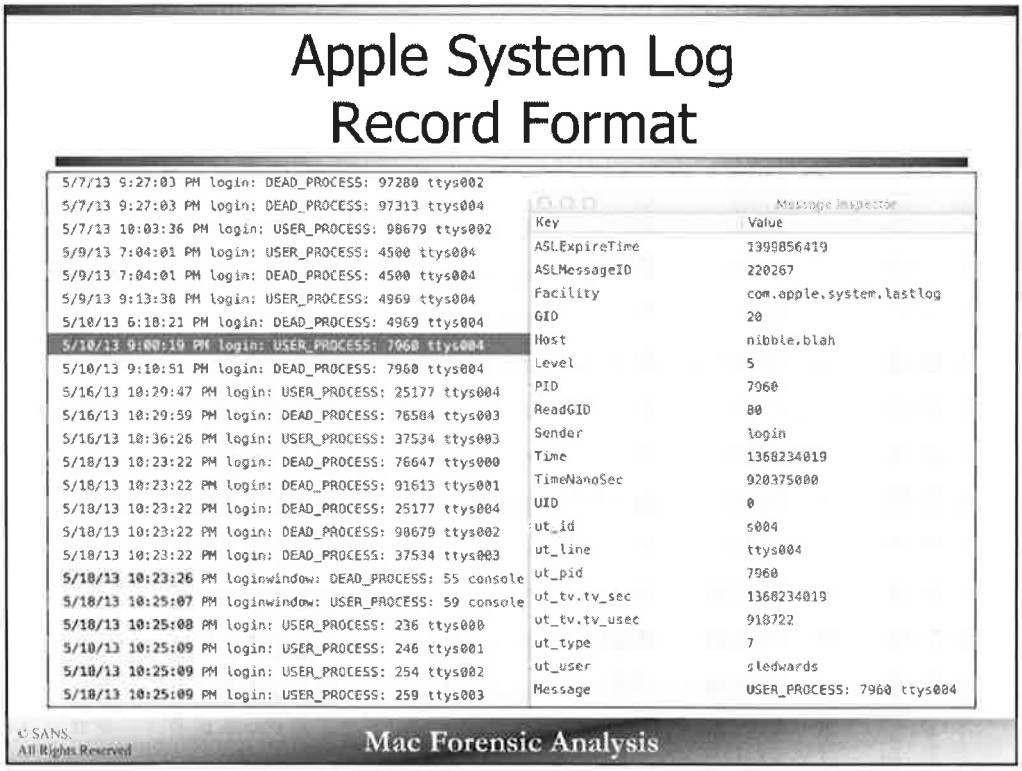
The "Best Before" log files contains the login records for the month listed in the filename. These records will be kept around for one year.

The directories starting with `AUX` or "Auxiliary" are used on 10.8+ systems. These directories contain text files comprising 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.



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

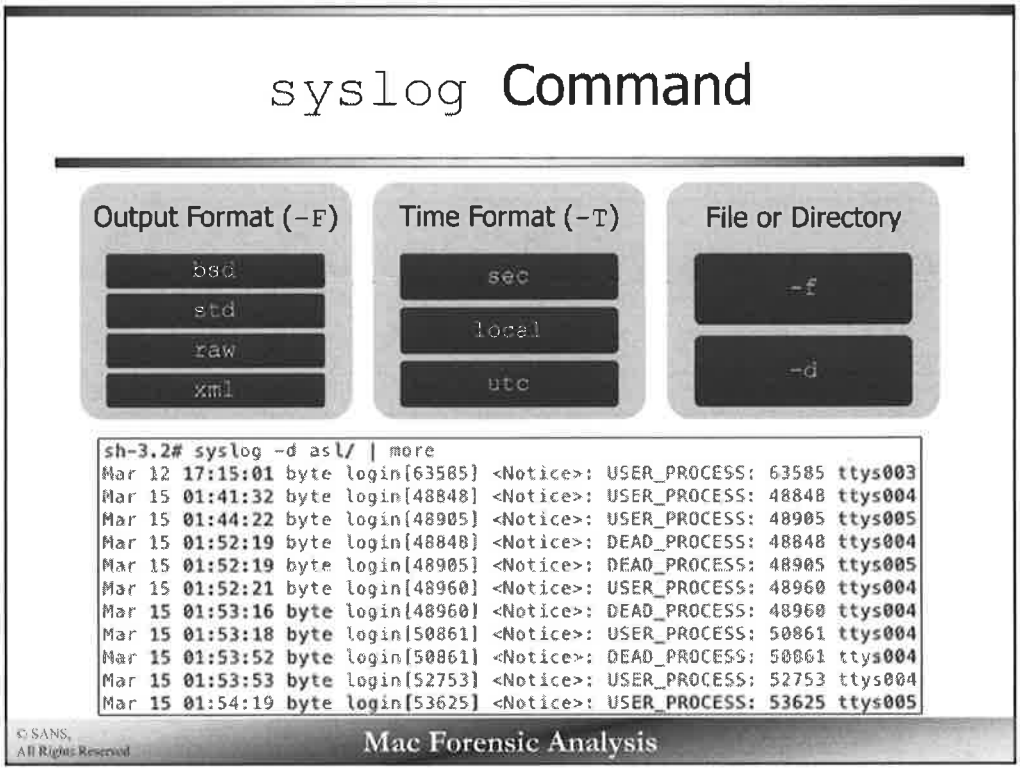

All Messages
Hide Log List
Clear Display
Reload
Ignore Sender
Insert Marker
Inspector
String Matching
Filter
SYSTEM LOG QUERIES
All Messages
DIAGNOSTIC AND USAGE...
Diagnostic and Usage Me...
User Diagnostic Reports
System Diagnostic Reports
FILES
system.log
~/Library/Logs
/Library/Logs
/var/log
alf.log
apache2
Senders
Tags
sandboxd
mdworker
kernel
UserEventAgent
Dock
...ple.backupd
WindowServer
configd
...onderHelper
airportd
8:31:08 PM sandboxd: ([39846]) mdworker(39846) deny mach-lookup com.apple.ls.boxd
8:31:08 PM sandboxd: ([39848]) mdworker(39848) deny mach-lookup com.apple.ls.boxd
8:31:08 PM sandboxd:
([39849]) mdworker(39849) deny mach-lookup com.apple.ls.boxd
Full Report
311804 — Locked
mdworker(39849) deny mach-lookup com.apple.ls.boxd
Process: mdworker [39849]
Path: /System/Library/Frameworks/CoreServices.framework/Frameworks/Metadata.framework/Versions/A/Support/mdworker
Load Address: 0x101e1b000
Identifier: mdworker
Version: ??? (???)
Code Type: x86_64 (Native)
Parent Process: launchd [153]
Date/Time: 2013-06-03 20:31:08.143 -0400
OS Version: Mac OS X 10.8.3 (12D78)
Report Version: 8
Thread 0:
0 libsystem_kernel.dylib 0x00007fff85683686 mach_msg_trap + 10
1 CoreFoundation 0x00007fff850a6233 __CFRunLoopServiceMachPort + 195
2 CoreFoundation 0x00007fff850ab916 __CFRunLoopRun + 1078
3 CoreFoundation 0x00007fff850ab0e2 CFRunLoopRunSpecific + 290
4 CoreFoundation 0x00007fff850b9dd1 CFRunLoopRun + 97
5 mdworker 0x0000000101e23ba9
6 libdyld.dylib 0x00007fff856937e1 start + 0
Thread 1:
0 libsystem_kernel.dylib 0x00007fff85685d16 kevent + 10
1 libdispatch.dylib 0x00007fff885219ee _dispatch_mgr_thread + 54
Thread 2:
0 libsystem_kernel.dylib 0x00007fff856836c2 semaphore_wait_trap + 10
4001 messages from 6/2/13 9:36:52 PM to 6/3/13 8:54:12 PM
Earlier
Later
Now



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

- **ASLExpireTime** – Message Expire Timestamp, this is 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.

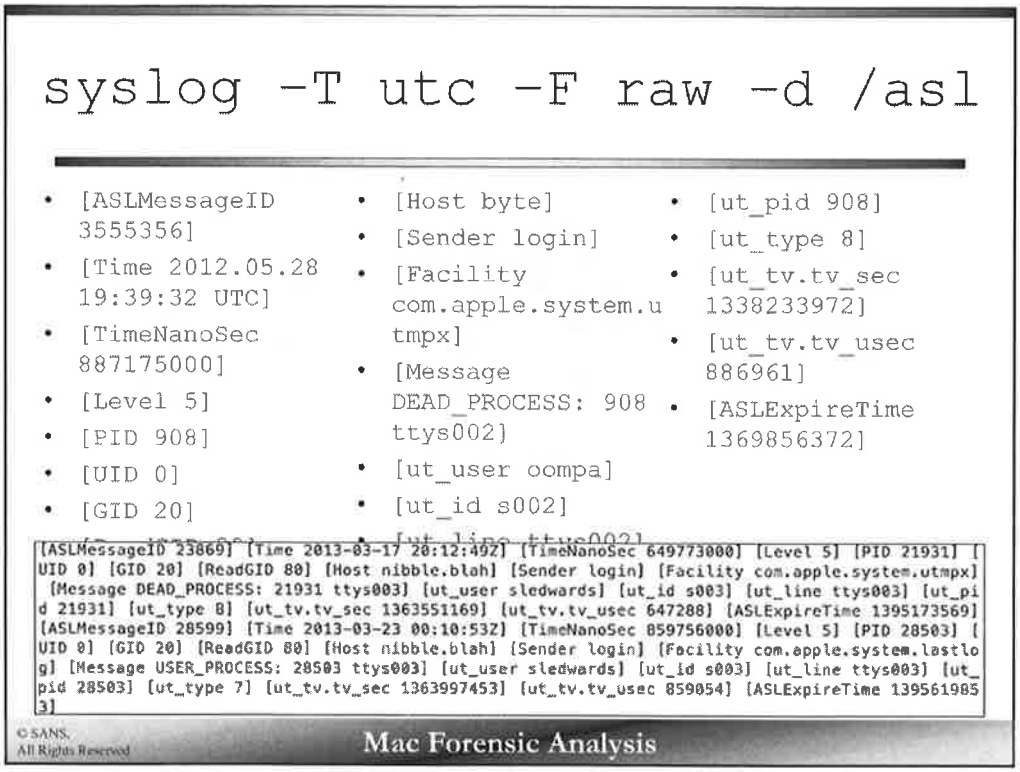
The syslog command can be used to output the data in a variety of formats:

- `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:

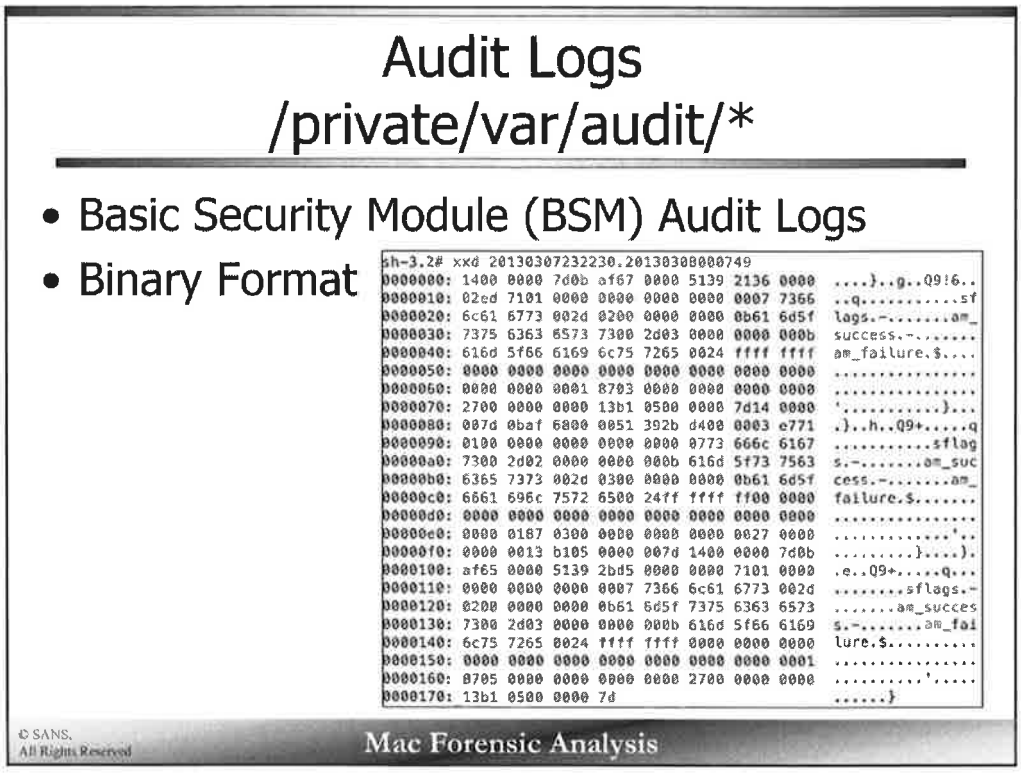
- `sec` – Number of seconds since epoch
- `local` – Local time zone (default)
- `utc` – UTC format

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.



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]
```



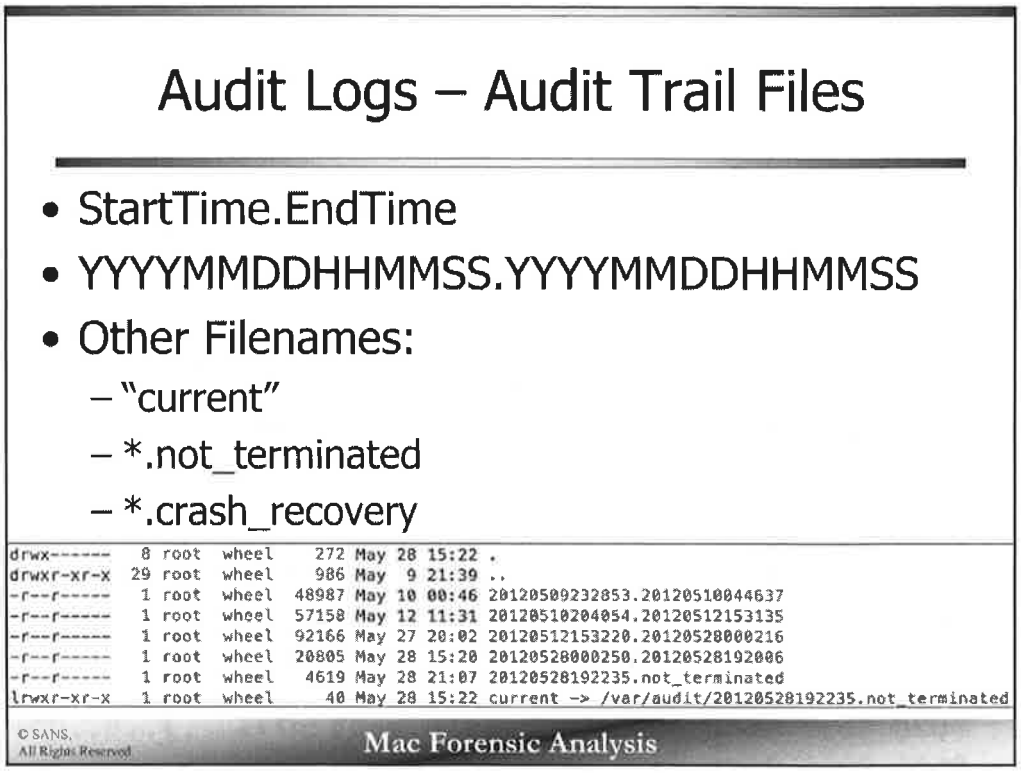
The audit logs are one of the few logs not located in the main log directories. These logs are located in the `/var/audit/` directory.

The audit logs use the Basic Security Module from OpenBSM (McAfee Research) based upon 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                        ......}
```

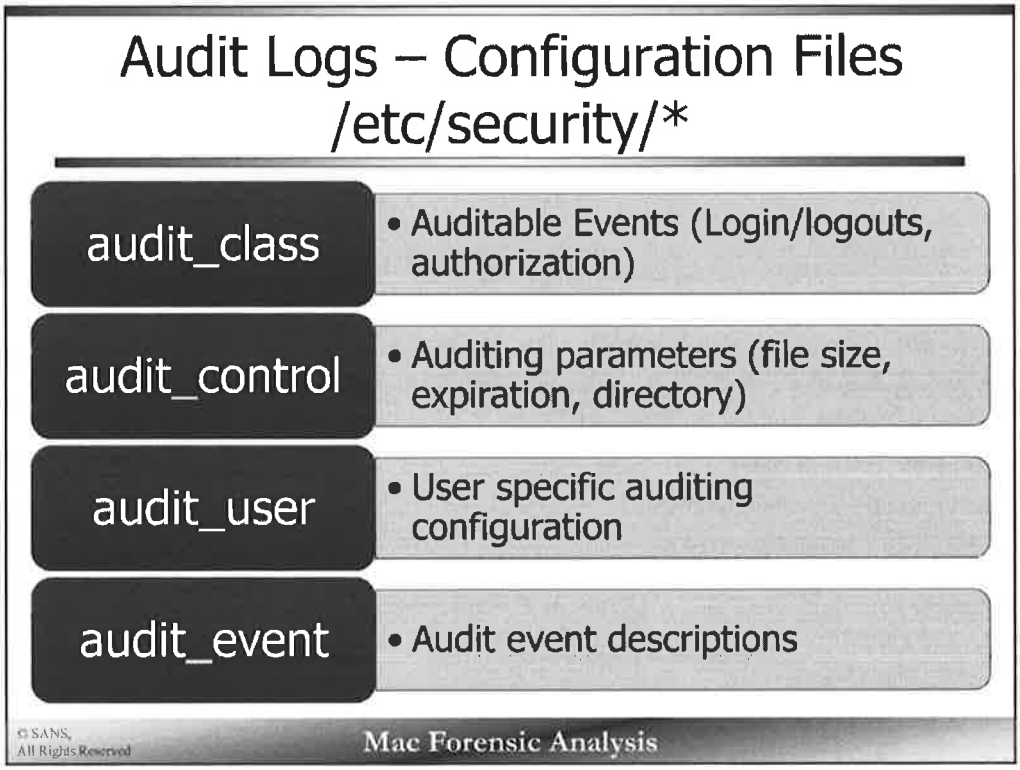


The log directory should contain many log files, each with a standard naming scheme (`StartTime.EndTime`) in the format `YYYYMMDDHHMMSS`. In the screenshot, the first log file (`20120509232853.20120510044637`) would contain data for the time period in between 05/09/2012 23:28:53 to 05/10/2012 04:64:37. Other file names 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. The screenshot above shows the default configuration for a 10.8 system:

- **`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`)

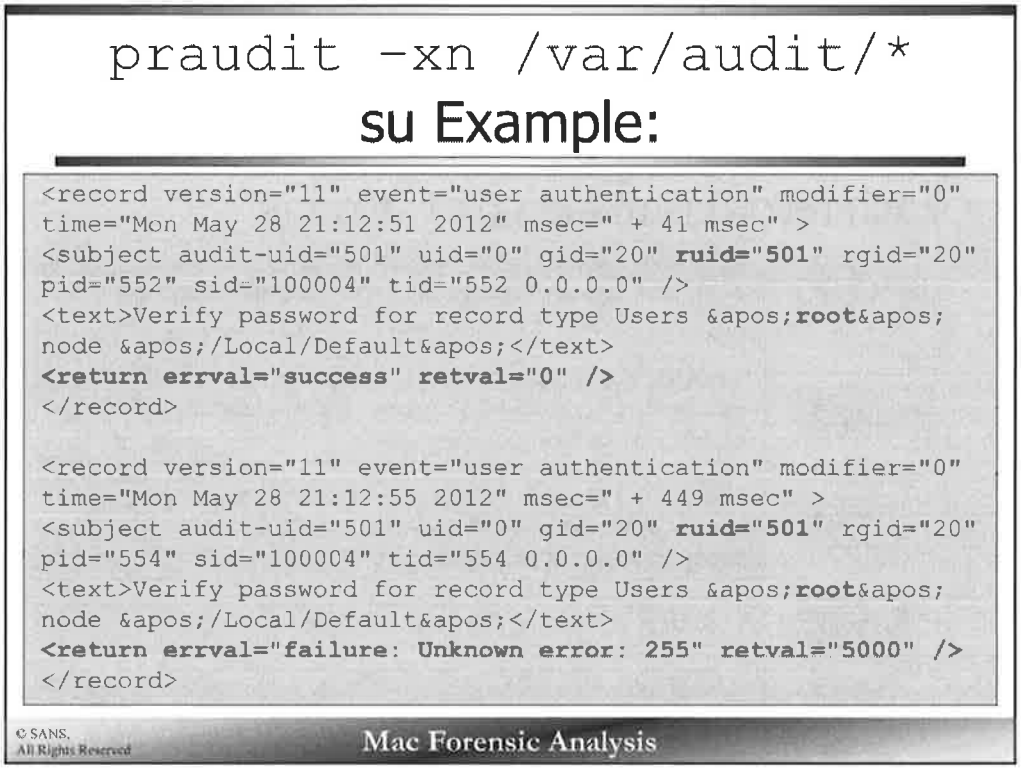
References:

`audit_class` Man Page

`audit_control` Man Page

`audit_user` Man Page

`audit_event` Man Page



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

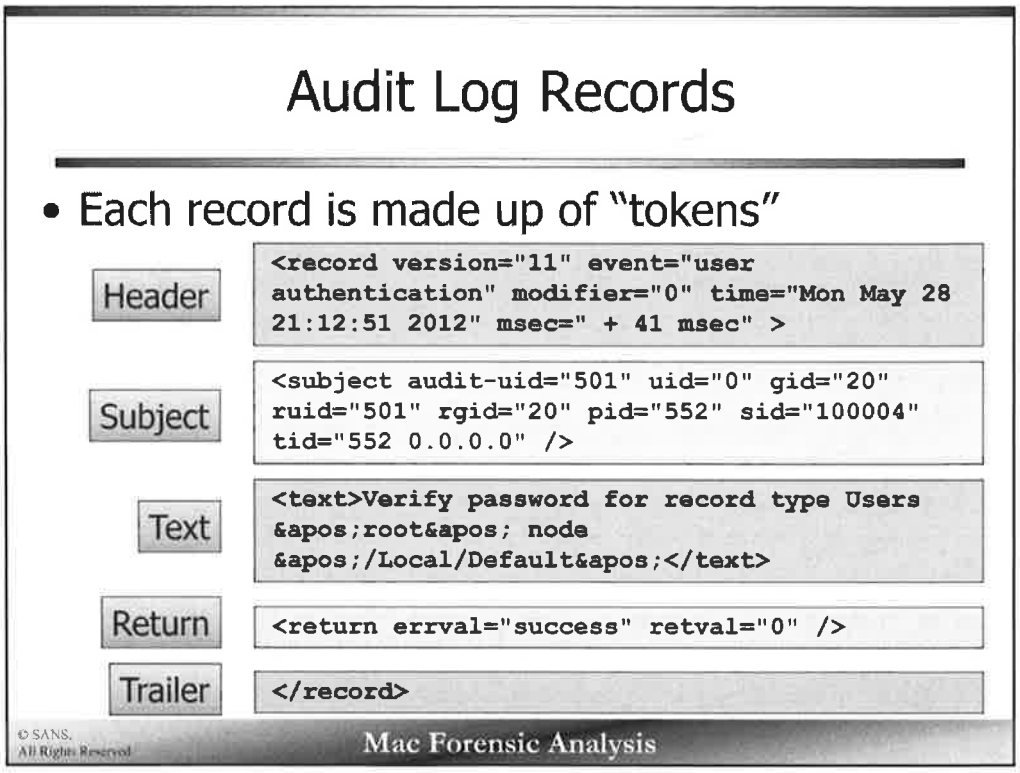
The `praudit` command shown above uses the `-xn` options to print the audit records in XML format (`-x`) and does not convert the user and group IDs (`-n`).

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.

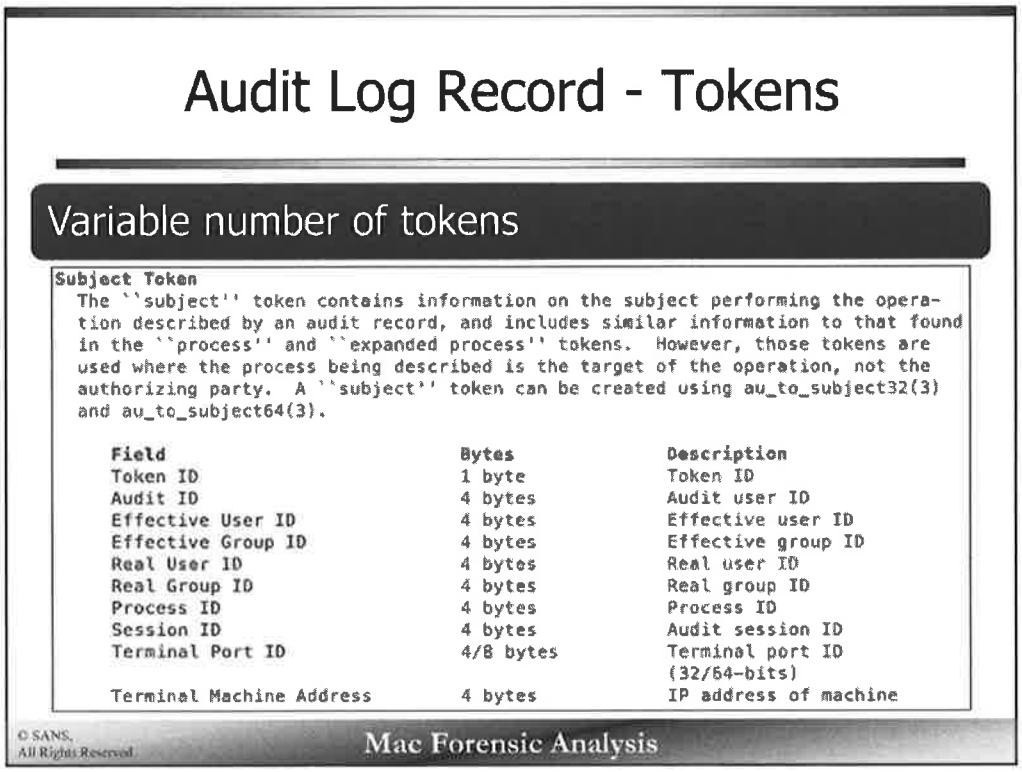
In the second record the same event occurred, but returned with a "`failure:Unknown error: 255`" error. This error type is returned at a failed '`su`' login.



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. 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 doing executing 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.

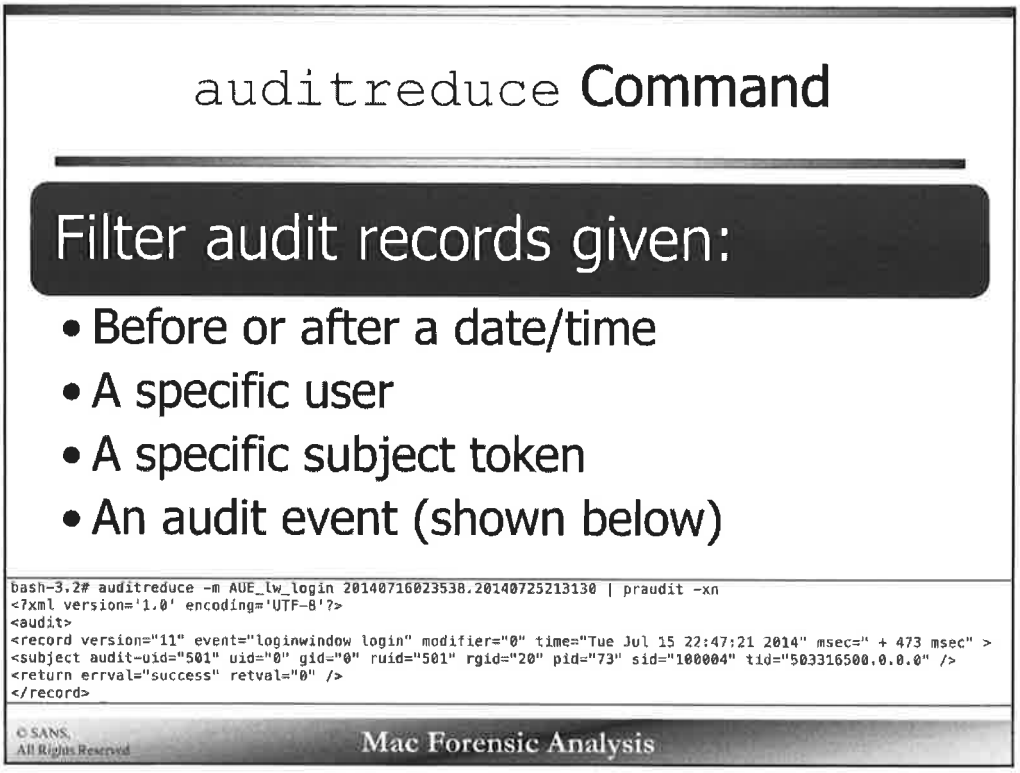


More information about each token can be found on the `audit.log` man page.

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

References:

`audit.log` Man Page



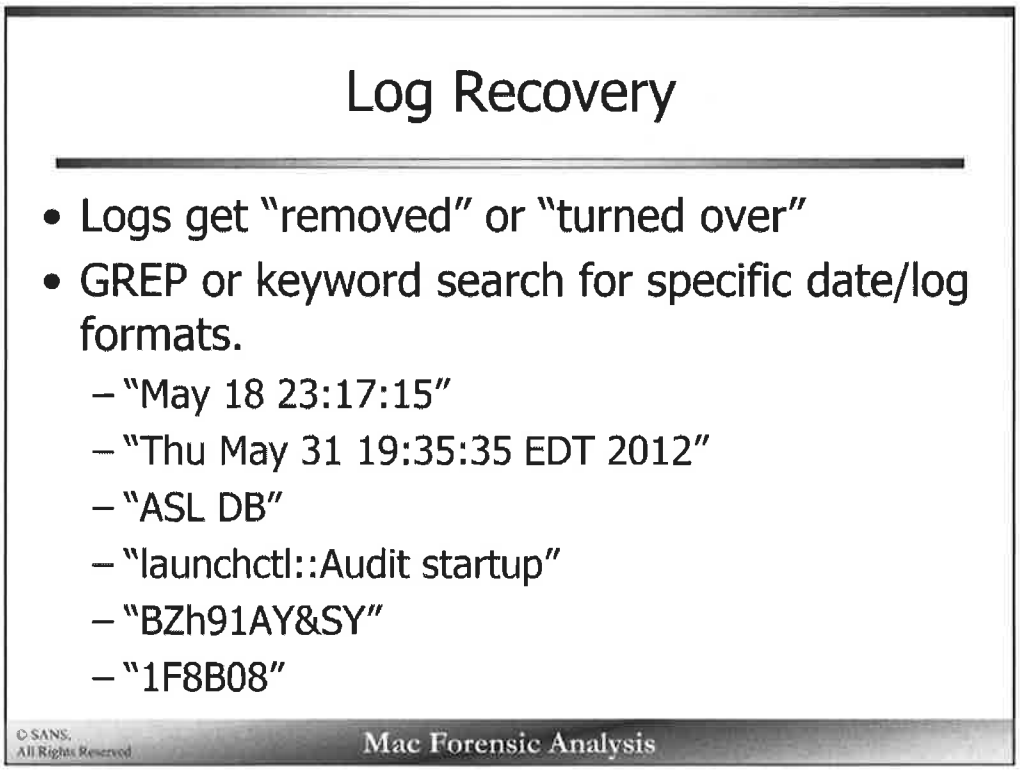
BSM audit records can be filtered using the `auditreduce` command.

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`.

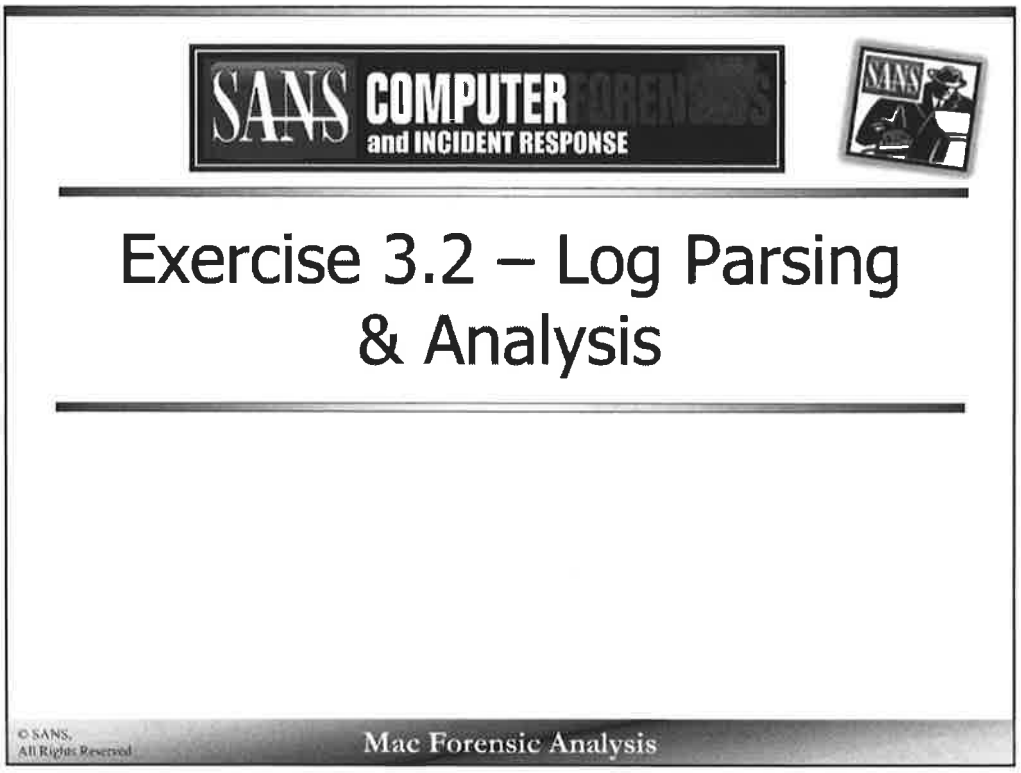
References:

`auditreduce` Man Page

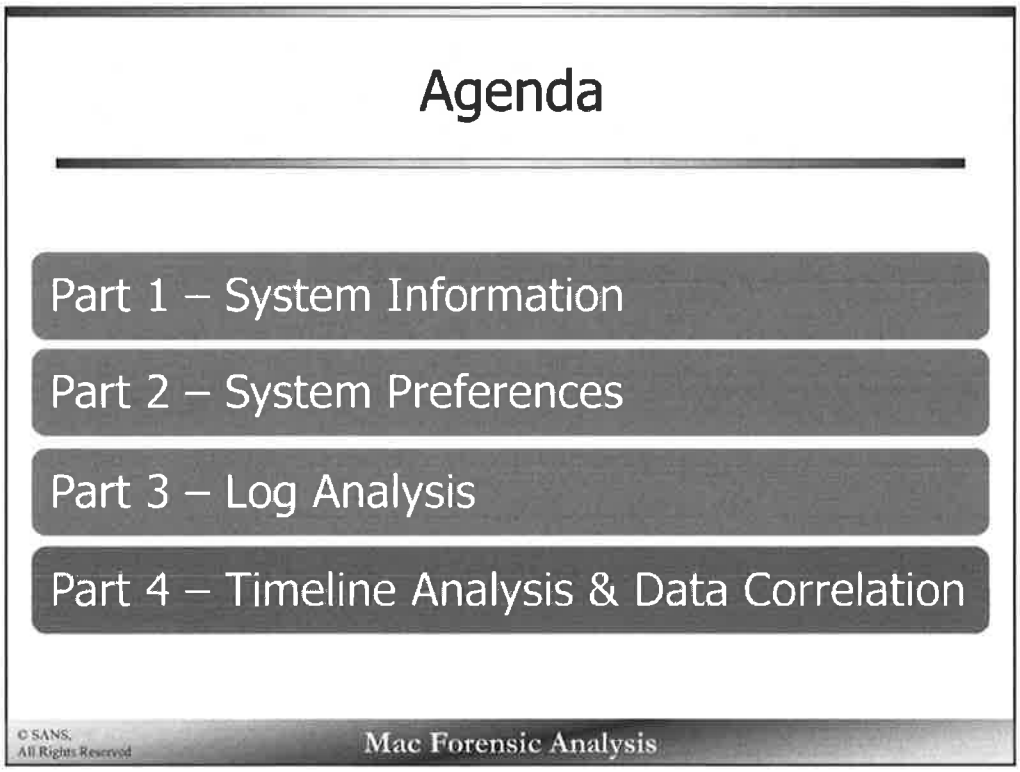


In the eventuality that log files have been archived or removed, they may be able to be recovered.

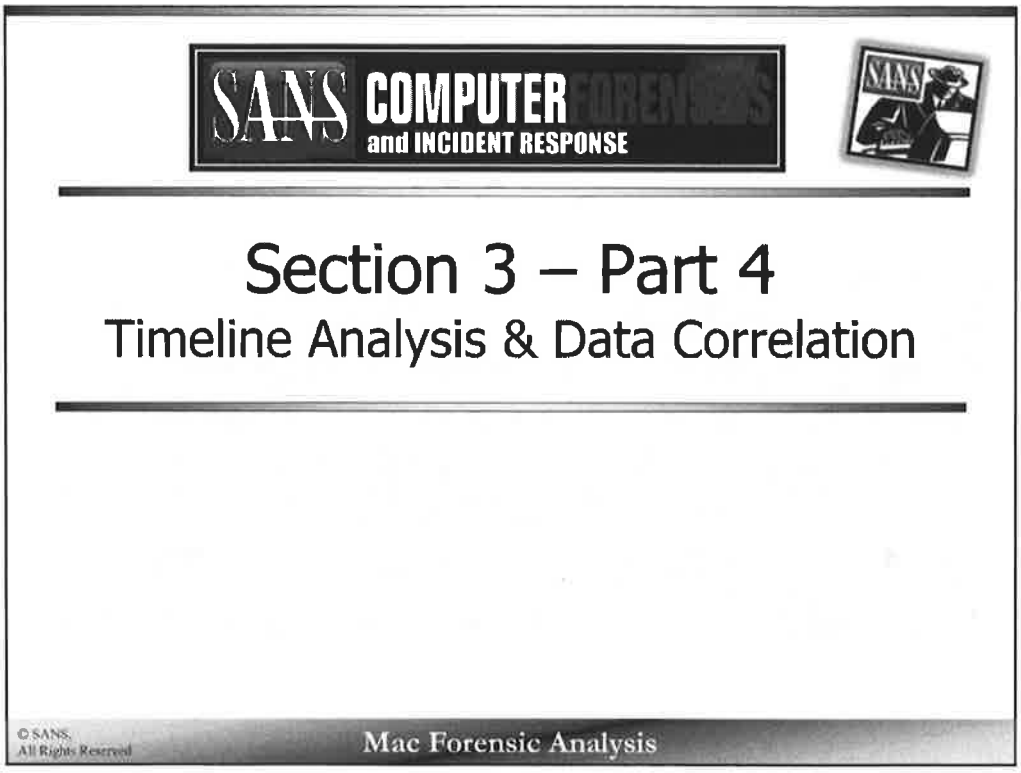
- A `grep` search for date formats of the specific log you are looking for may be helpful.
- Log signatures for binary logs such as "`ASL DB`" for the Apple System Log can be searched for.
- For archived files, the BZip2 signature should be searched, "`BZh91AY&SY`"
- For gzipped archived files, use "`1F8B08`".



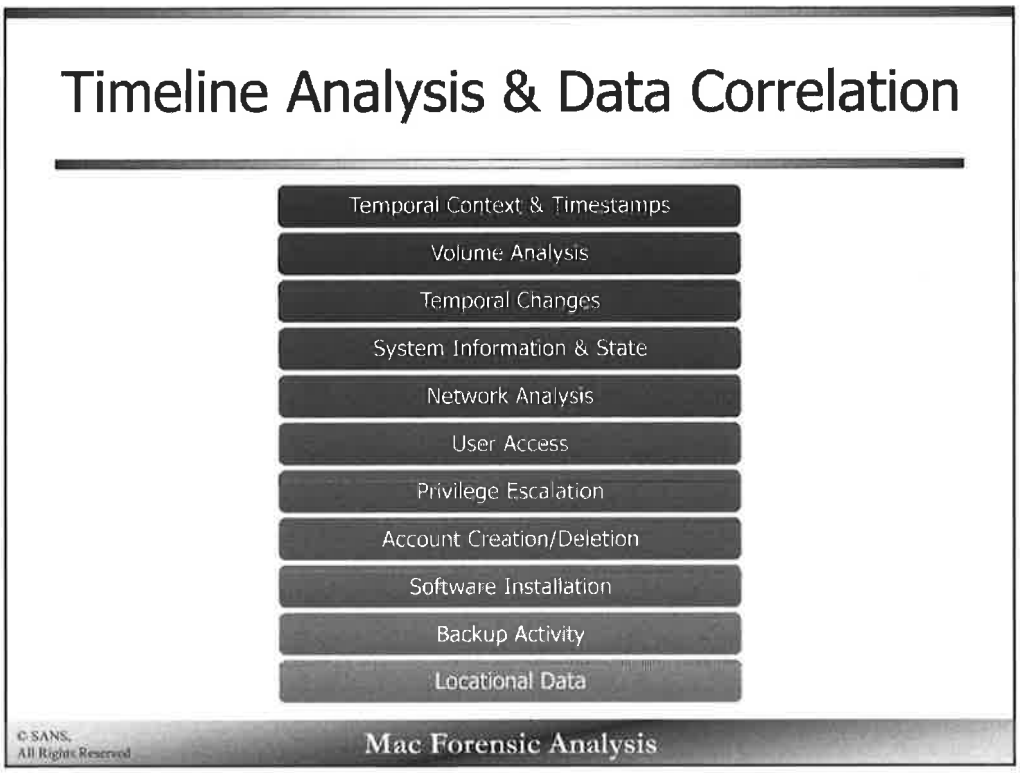
This page intentionally left blank.



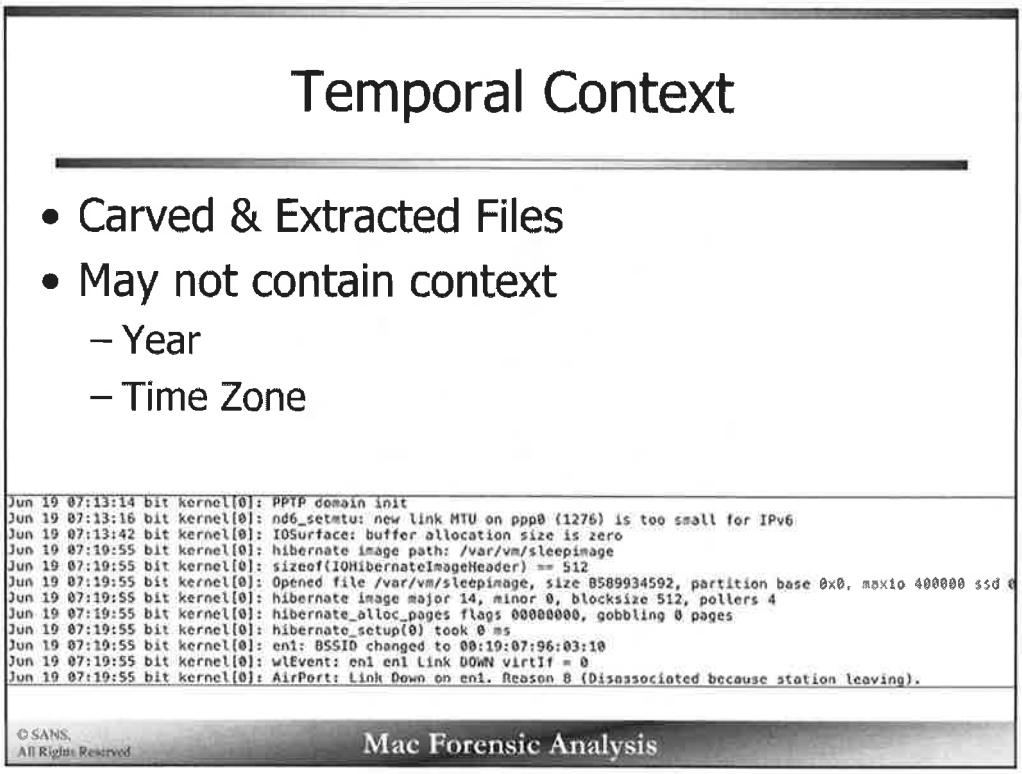
This page intentionally left blank.



This page intentionally left blank.



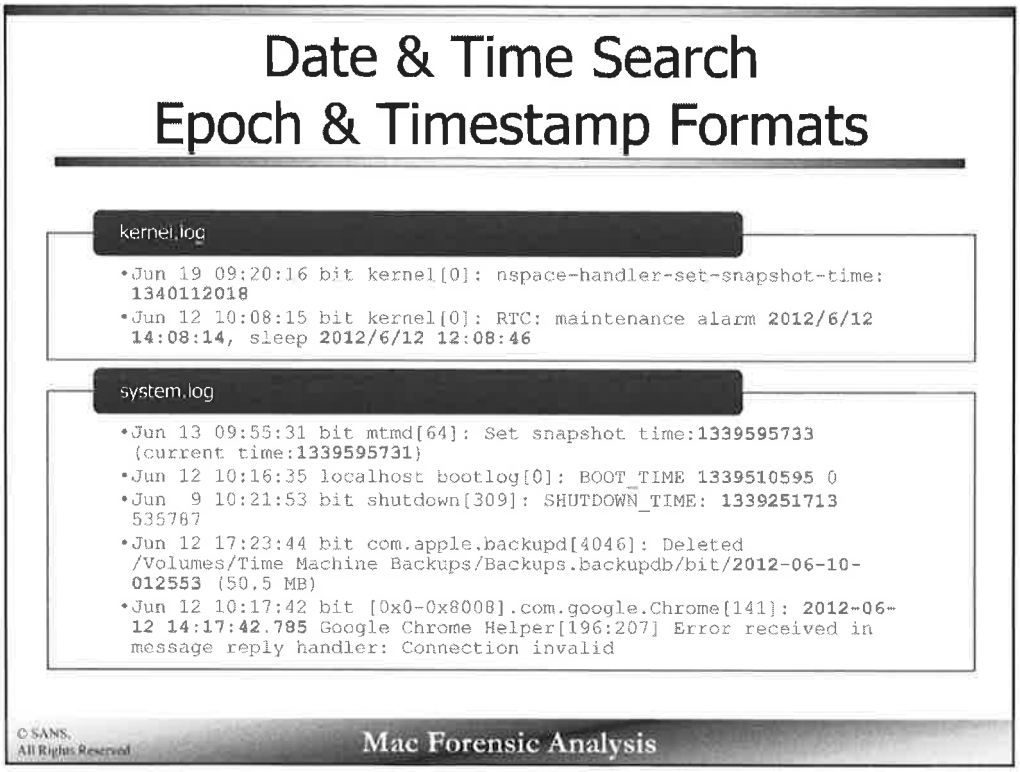
This page intentionally left blank.



Most of the log files on OS X are based upon standard Unix log formats, which do not always contain the year the event happened. While the MAC times of a log file may provide that information to current log files, files extracted from unallocated space do not have these timestamps.

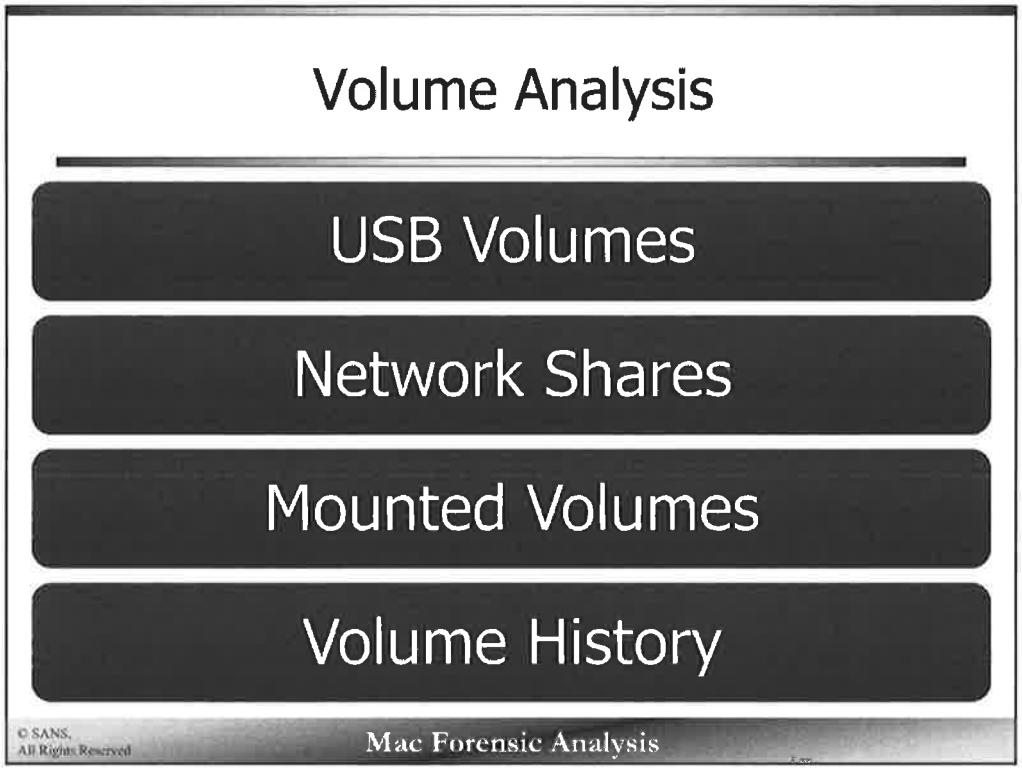
The example shows a log that contains some entries which might be of interest. These entries happened on June 19th, but we do not know what year. We can look at the content of a particular entry to gain perspective as to what year the event likely occurred. Viewing the log entries just before and just after the entries in question could be your best bet.

```
Jun 19 07:13:14 bit kernel[0]: PPTP domain init
Jun 19 07:13:16 bit kernel[0]: nd6_setmtu: new link MTU on ppp0 (1276) is too small for IPv6
Jun 19 07:13:42 bit kernel[0]: IOSurface: buffer allocation size is zero
Jun 19 07:19:55 bit kernel[0]: hibernate image path: /var/vm/sleepimage
Jun 19 07:19:55 bit kernel[0]: sizeof(IOHibernateImageHeader) == 512
Jun 19 07:19:55 bit kernel[0]: Opened file /var/vm/sleepimage, size 8589934592, partition base 0x0, maxio 400000 ssd 0
Jun 19 07:19:55 bit kernel[0]: hibernate image major 14, minor 0, blocksize 512, pollers 4
Jun 19 07:19:55 bit kernel[0]: hibernate_alloc_pages flags 00000000, gobbling 0 pages
Jun 19 07:19:55 bit kernel[0]: hibernate_setup(0) took 0 ms
Jun 19 07:19:55 bit kernel[0]: en1: BSSID changed to 00:19:07:96:03:10
Jun 19 07:19:55 bit kernel[0]: wlEvent: en1 en1 Link DOWN virtIf = 0
Jun 19 07:19:55 bit kernel[0]: AirPort: Link Down on en1. Reason 8 (Disassociated because station leaving).
```

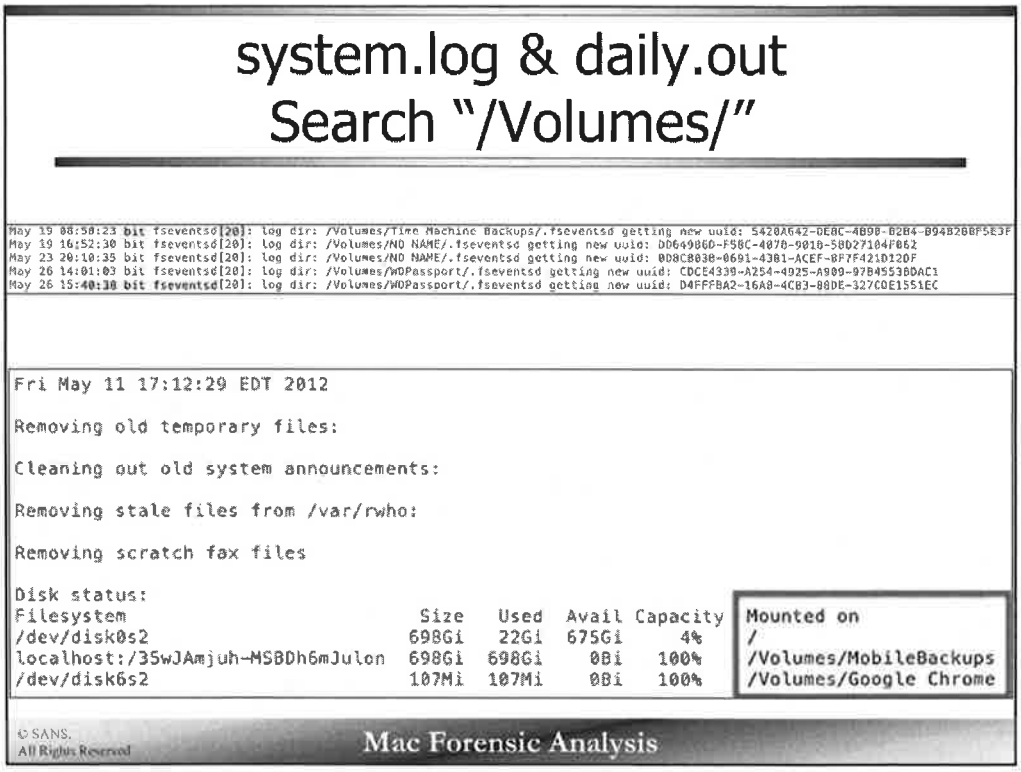


Many dates and times are stored in the actual message of the log entry, in many different timestamp formats. An analyst needs to be familiar with these timestamps so they can put context around an unknown event. These date formats can be searched within the unallocated space to find entries that happened on a date of interest for a specific case. Each timestamp type should be searched to find the log data.

The Unix epoch timestamps can be decoded into human readable format by using the `date -ur` command.



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



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.

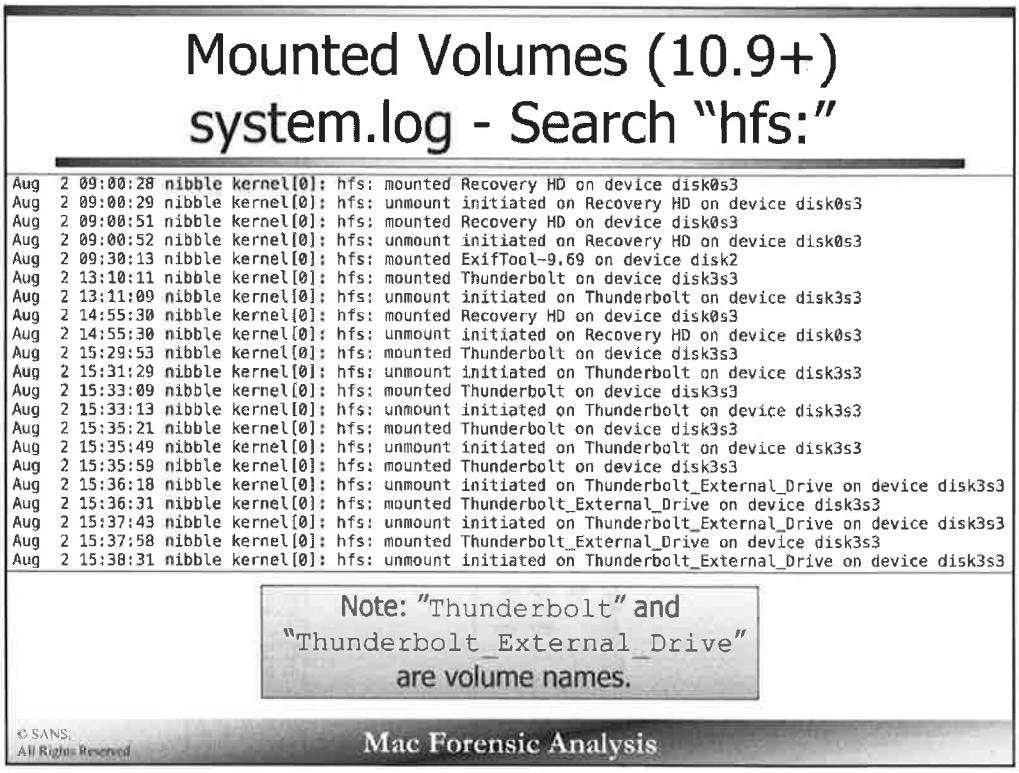
The term "`/Volumes/`" can be searched for in the `system.log` and `daily.out` files. The `/Volumes/` directory is the default mount point for any volume that is mounted on the system.

The `system.log` file shows when the volume was automatically mounted and the UUID attached to that volume. It is important to point out that the same USB drive will have a different UUID every time it is mounted on the system.

The `daily.out` file shows what volumes were mounted on the system when the daily maintenance script is 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.

```
Fri May 11 17:12:29 EDT 2012

Removing old temporary files:

Cleaning out old system announcements:

Removing stale files from /var/rwho:

Removing scratch fax files

Disk status:
Filesystem                          Size   Used  Avail Capacity  Mounted on
/dev/disk0s2                       698Gi   22Gi  675Gi     4%    /
localhost:/35wJAmjuh-MSBDh6mJulon  698Gi  698Gi    0Bi   100%    /Volumes/MobileBackups
/dev/disk6s2                       107Mi  107Mi    0Bi   100%    /Volumes/Google Chrome
```

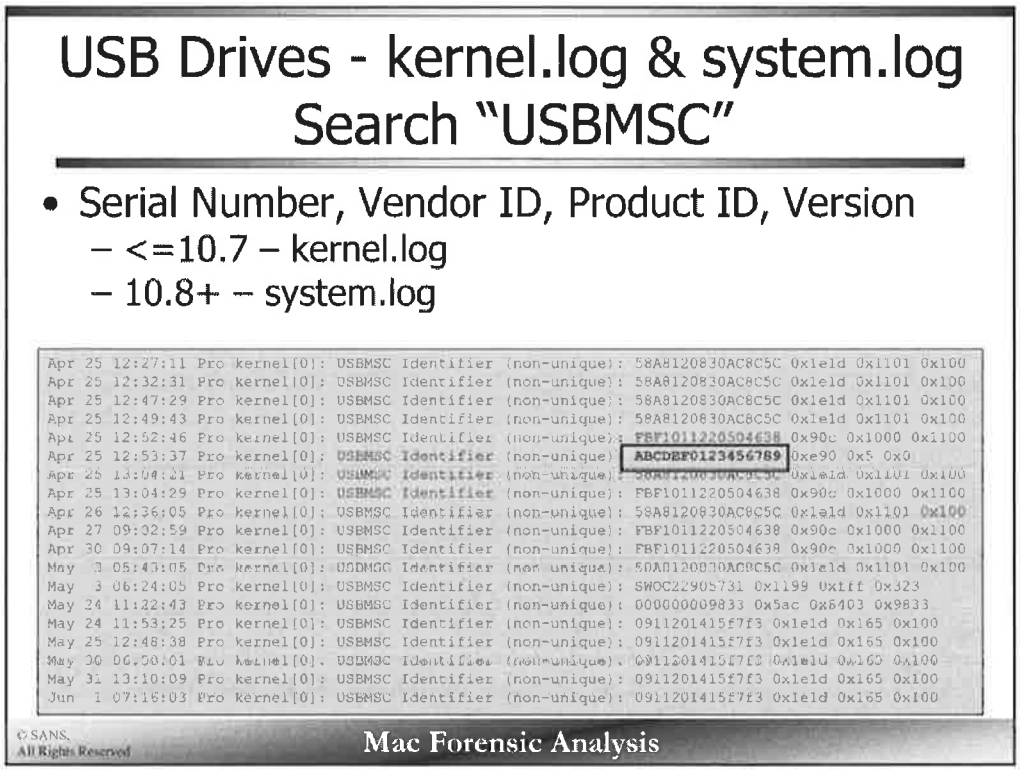
```
May 19 08:58:23 bit fseventsd[20]: log dir: /Volumes/Time Machine Backups/.fseventsd getting new uuid: 5420A642-DE8C-4B90-B2B4-B948288F5E3F
May 19 16:52:30 bit fseventsd[20]: log dir: /Volumes/NO NAME/.fseventsd getting new uuid: DD64986D-F58C-407B-9018-5BD27104F062
May 23 20:10:35 bit fseventsd[20]: log dir: /Volumes/NO NAME/.fseventsd getting new uuid: 0D8CB03B-0691-4381-ACEF-8F7F421D12DF
May 26 14:01:03 bit fseventsd[20]: log dir: /Volumes/WDPassport/.fseventsd getting new uuid: CDCE4339-A254-4925-A909-97B4553BDAC1
May 26 15:40:38 bit fseventsd[20]: log dir: /Volumes/WDPassport/.fseventsd getting new uuid: D4FFFBA2-16A8-4CB3-88DE-327CDE1551EC
```



10.9 systems show when volumes were mounted and unmounted.

These records can be found in the `system.log` by searching for "`hfs:`", "`mounted`", or "`unmounted`".

They records show the `/dev` mount point as well as the volume name (i.e.,: "ExifTool-9.96" or "Thunderbolt").



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 `kernel.log` on systems using 10.7 and earlier. With 10.8+, the kernel log data has been incorporated into the `system.log`.

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 0-9 as its identifier. The additional numbers are the vendor ID, Product ID, and version.

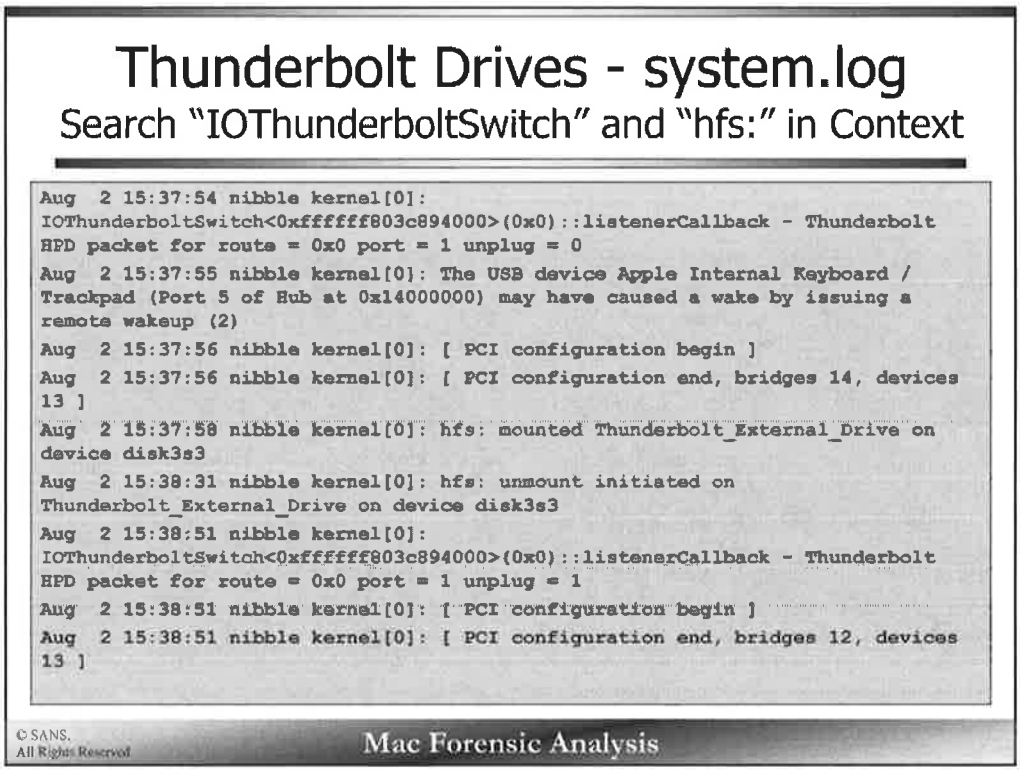
```
Jun  3 11:11:53 bit kernel[0]: USBMSC Identifier (non-unique):
FBF1011220504638 0x90c 0x1000 0x1100
```

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

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

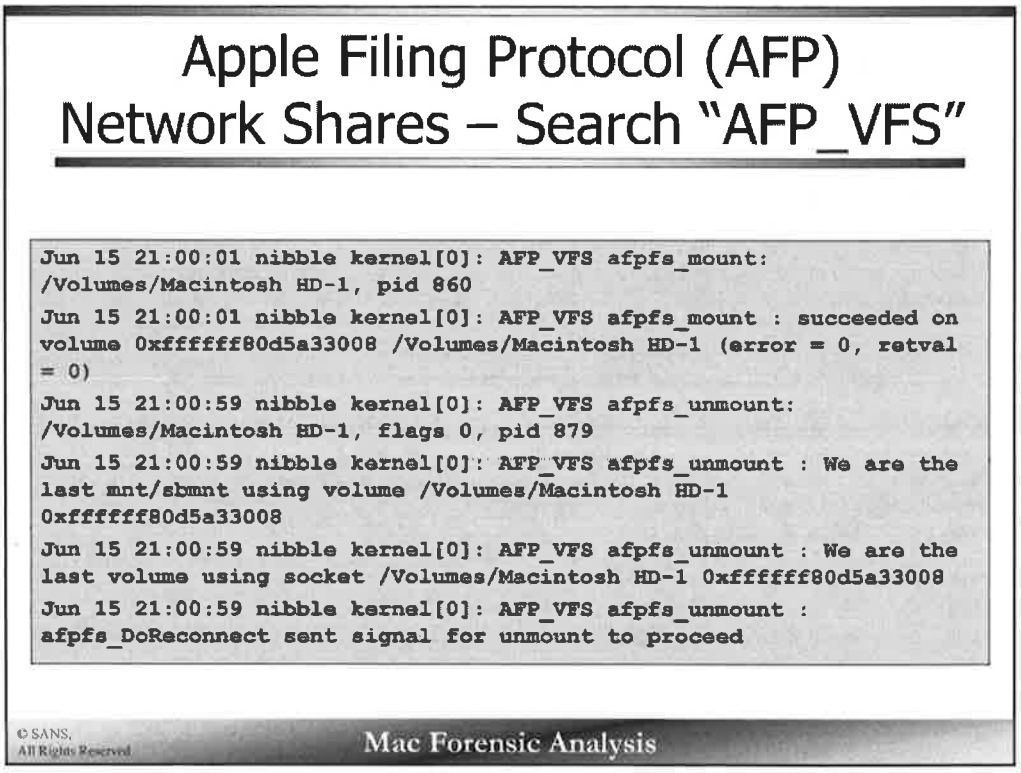
The website: http://usb-ids.gowdy.us/read/UD/ lists vendors by ID and products by ID. It may also link to a forum discussing the physical properties of the device.

```
Apr 25 12:27:11 Pro kernel[0]: USBMSC Identifier (non-unique):
58A8120830AC8C5C 0x1e1d 0x1101 0x100

Apr 25 12:32:31 Pro kernel[0]: USBMSC Identifier (non-unique):
58A8120830AC8C5C 0x1e1d 0x1101 0x100

Apr 25 12:47:29 Pro kernel[0]: USBMSC Identifier (non-unique):
58A8120830AC8C5C 0x1e1d 0x1101 0x100

Apr 25 12:49:43 Pro kernel[0]: USBMSC Identifier (non-unique):
58A8120830AC8C5C 0x1e1d 0x1101 0x100

Apr 25 12:52:46 Pro kernel[0]: USBMSC Identifier (non-unique):
FBF1011220504638 0x90c 0x1000 0x1100

Apr 25 12:53:37 Pro kernel[0]: USBMSC Identifier (non-unique):
ABCDEF0123456789 0xe90 0x5 0x0

Apr 25 13:04:21 Pro kernel[0]: USBMSC Identifier (non-unique):
58A8120830AC8C5C 0x1e1d 0x1101 0x100

Apr 25 13:04:29 Pro kernel[0]: USBMSC Identifier (non-unique):
FBF1011220504638 0x90c 0x1000 0x1100

Apr 26 12:36:05 Pro kernel[0]: USBMSC Identifier (non-unique):
58A8120830AC8C5C 0x1e1d 0x1101 0x100

Apr 27 09:02:59 Pro kernel[0]: USBMSC Identifier (non-unique):
FBF1011220504638 0x90c 0x1000 0x1100

Apr 30 09:07:14 Pro kernel[0]: USBMSC Identifier (non-unique):
FBF1011220504638 0x90c 0x1000 0x1100

May  3 05:43:05 Pro kernel[0]: USBMSC Identifier (non-unique):
58A8120830AC8C5C 0x1e1d 0x1101 0x100

May  3 06:24:05 Pro kernel[0]: USBMSC Identifier (non-unique): SWOC22905731
0x1199 0xfff 0x323

May 24 11:22:43 Pro kernel[0]: USBMSC Identifier (non-unique): 000000009833
0x5ac 0x8403 0x9833

May 24 11:53:25 Pro kernel[0]: USBMSC Identifier (non-unique):
0911201415f7f3 0x1e1d 0x165 0x100

May 25 12:48:38 Pro kernel[0]: USBMSC Identifier (non-unique):
0911201415f7f3 0x1e1d 0x165 0x100

May 30 06:50:01 Pro kernel[0]: USBMSC Identifier (non-unique):
0911201415f7f3 0x1e1d 0x165 0x100

May 31 13:10:09 Pro kernel[0]: USBMSC Identifier (non-unique):
0911201415f7f3 0x1e1d 0x165 0x100

Jun  1 07:16:03 Pro kernel[0]: USBMSC Identifier (non-unique):
0911201415f7f3 0x1e1d 0x165 0x100
```



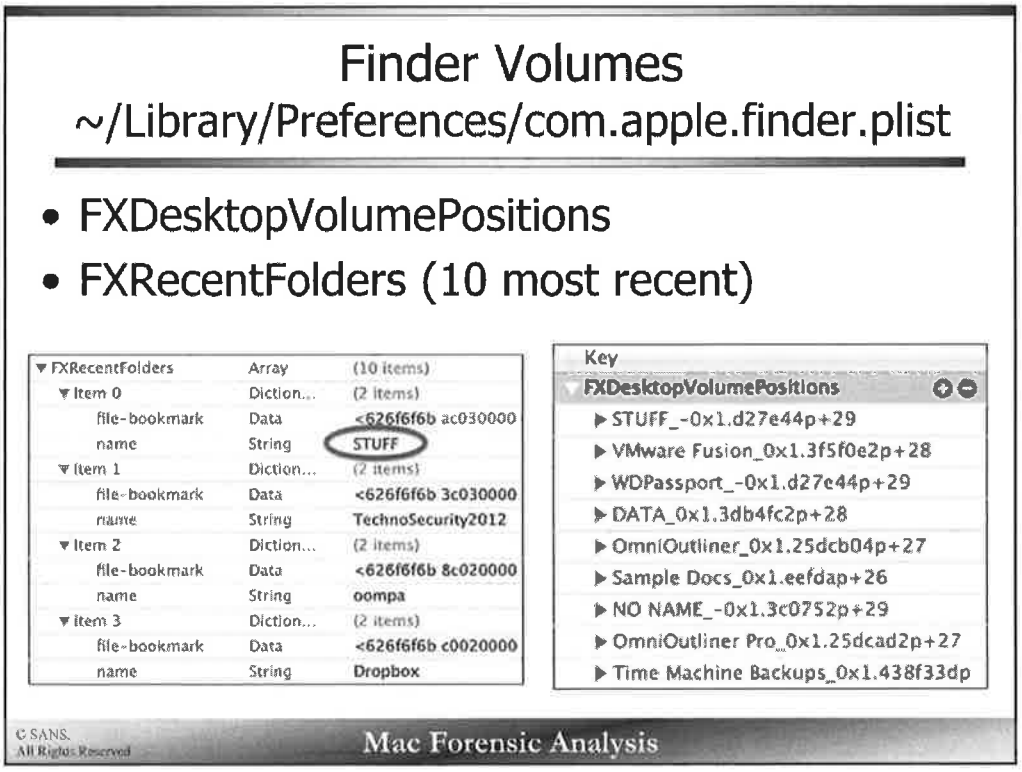
The log records shown a Thunderbolt external drive plugged into a system, mounted, unmounted, and removed from the system.

The logs with the "hfs:" messages show the name of the mounted volume on the Thunderbolt drive – "`Thunderbolt_External_Drive`".



When an Apple Filing Protocol (AFP) share is accessed, such as those created via the Sharing preferences, you'll see these shares accessed in the `kernel.log` (<=10.7) or `system.log` (10.8+) files.

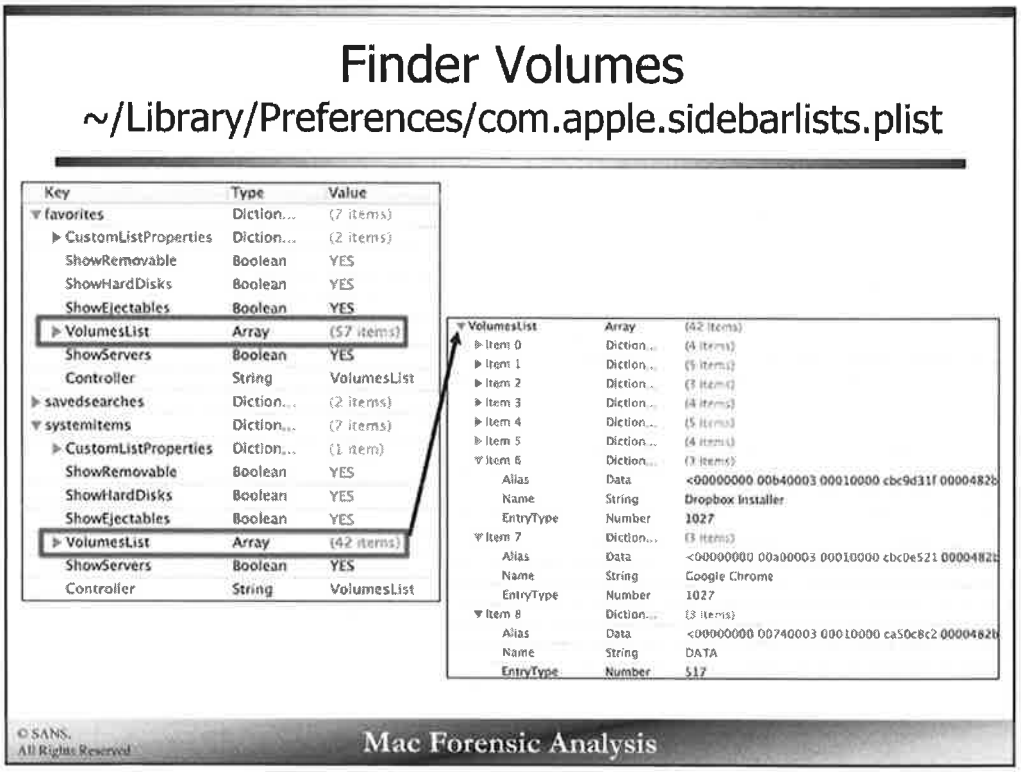
Each entry related to these shares can be found by searching for "`AFP_VFS`". The entry that shows the volume has been mounted contains an "`afpfs_mount`" keyword and the name and location of the mount. The entry that shows the share has been un-mounted contains the keyword "`afpfs_umount`" along with the name and location.



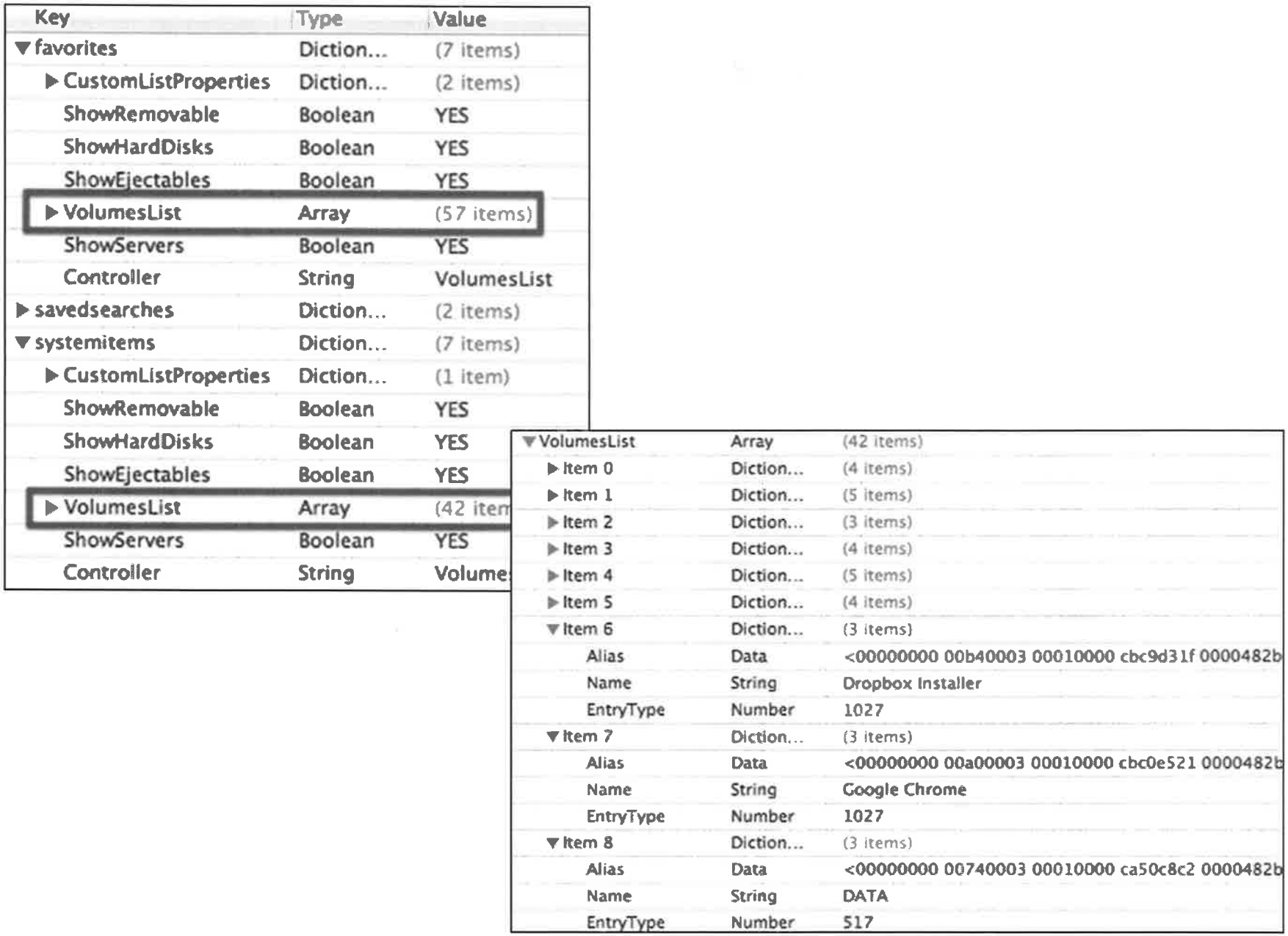
Each user will have their own plist files containing their volume history.

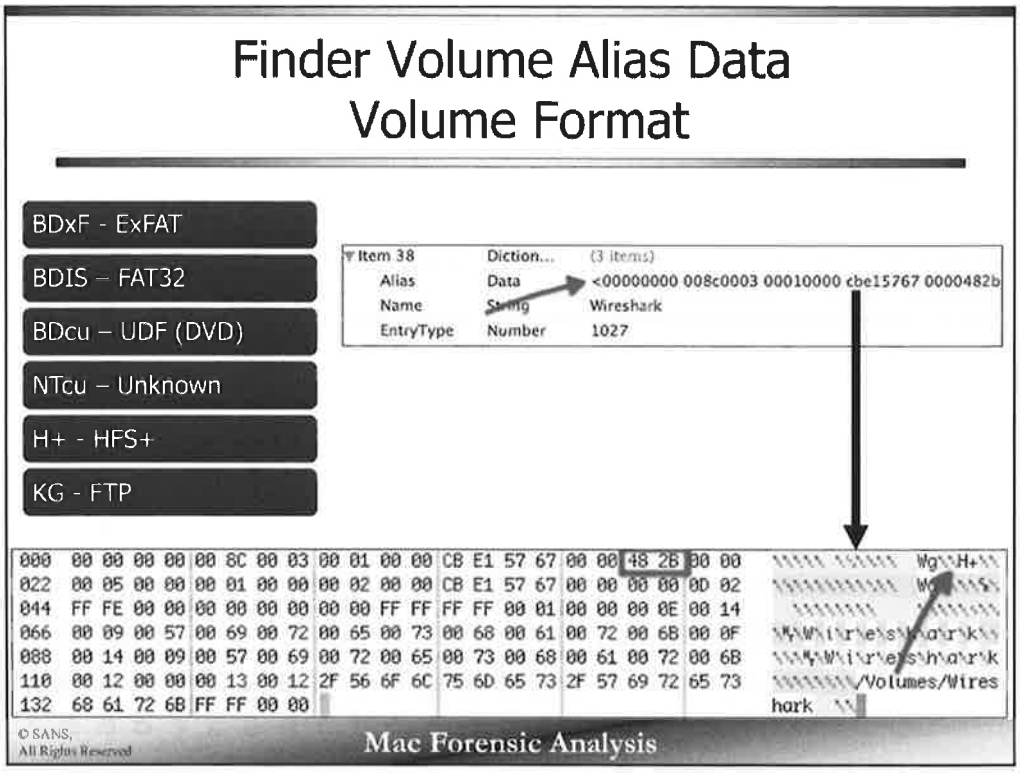
The `FXRecentFolders` key contains the volumes mounted on the system for the user along with a bookmark data blob. This blob data can be extracted and viewed in a hex editor or property list editor to gain more information about the volume.

The `FXDesktopVolumePositions` contains a list of volumes. Neither of these keys contain timestamps, therefore correlation with other data points will have to be done to determine when these volumes were available on the system.



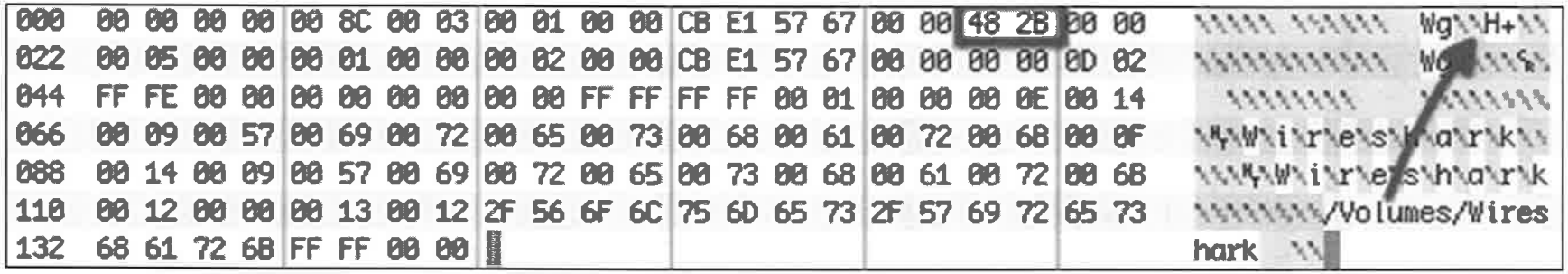
The general difference between the two `VolumeList` Keys is the items on the sidebar such as Applications, Documents, and Downloads.


Key Type Value
favorites Diction... (7 items)
CustomListProperties Diction... (2 items)
ShowRemovable Boolean YES
ShowHardDisks Boolean YES
ShowEjectables Boolean YES
VolumesList Array (57 items)
ShowServers Boolean YES
Controller String VolumesList
savedsearches Diction... (2 items)
systemitems Diction... (7 items)
CustomListProperties Diction... (1 item)
ShowRemovable Boolean YES
ShowHardDisks Boolean YES
ShowEjectables Boolean YES
VolumesList Array (42 item
ShowServers Boolean YES
Controller String Volume
VolumesList Array (42 items)
Item 0 Diction... (4 items)
Item 1 Diction... (5 items)
Item 2 Diction... (3 items)
Item 3 Diction... (4 items)
Item 4 Diction... (5 items)
Item 5 Diction... (4 items)
Item 6 Diction... (3 items)
Alias Data <00000000 00b40003 00010000 cbc9d31f 0000482b
Name String Dropbox Installer
EntryType Number 1027
Item 7 Diction... (3 items)
Alias Data <00000000 00a00003 00010000 cbc0e521 0000482b
Name String Google Chrome
EntryType Number 1027
Item 8 Diction... (3 items)
Alias Data <00000000 00740003 00010000 ca50c8c2 0000482b
Name String DATA
EntryType Number 517



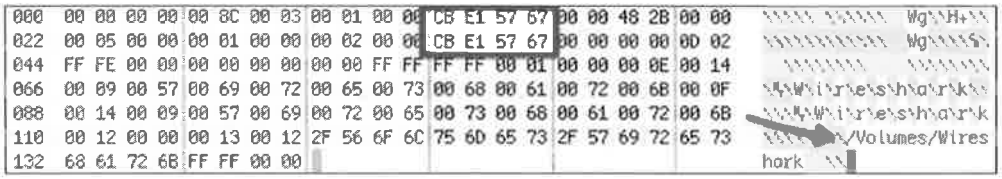
The alias blob data can be extracted and read in a hex editor. Each alias data blob contains a file system format, many of which are listed in the slide itself.

Some of these volume formats are currently unknown. Further testing will be required.

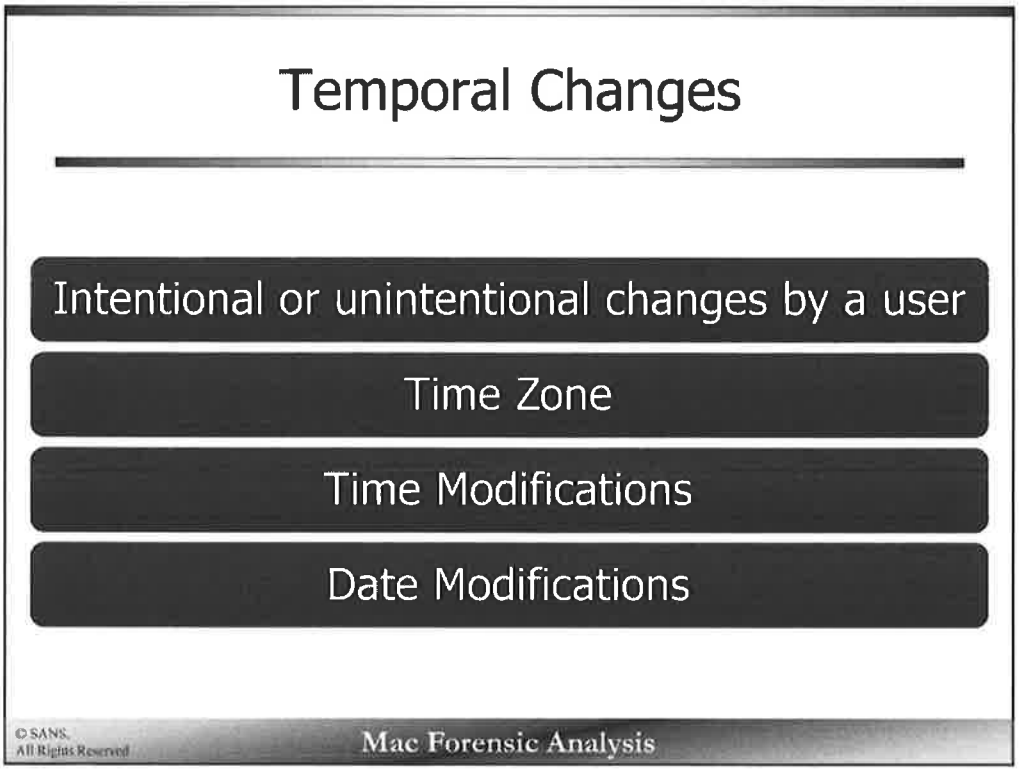

000 00 00 00 00 00 8C 00 03 00 01 00 00 CB E1 57 67 00 00 48 2B 00 00
022 00 05 00 00 00 01 00 00 00 02 00 00 CB E1 57 67 00 00 00 00 0D 02
044 FF FE 00 00 00 00 00 00 00 00 FF FF FF FF 00 01 00 00 00 0E 00 14
066 00 09 00 57 00 69 00 72 00 65 00 73 00 68 00 61 00 72 00 68 00 0F
088 00 14 00 09 00 57 00 69 00 72 00 65 00 73 00 68 00 61 00 72 00 68
110 00 12 00 00 00 13 00 12 2F 56 6F 6C 75 6D 65 73 2F 57 69 72 65 73
132 68 61 72 68 FF FF 00 00
/Volumes/Wires
hark

Finder Volume Alias Data Dates & Mount Point

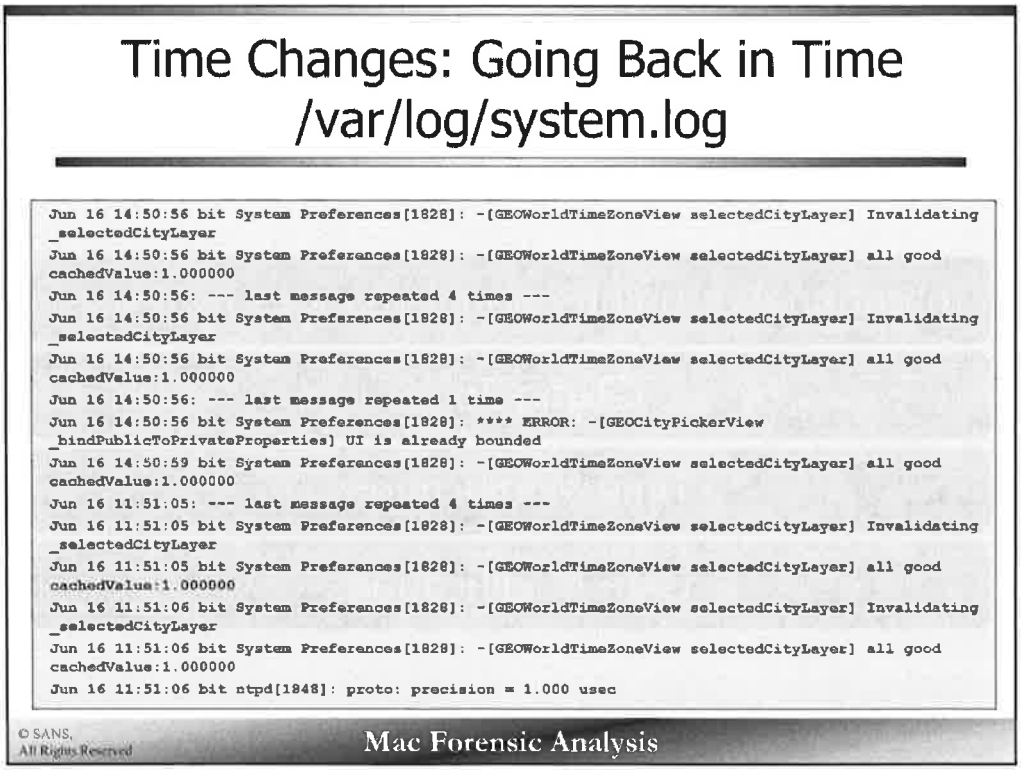
- Volume Name & Mount Point
- File Creation Time (HFS Date)
- May or may not be present
  - Only seen on H+ formatted disks
  - DMG



The alias blob data will also contain the volume name and mount point on the system. Creation dates may also be present on HFS+ volumes.



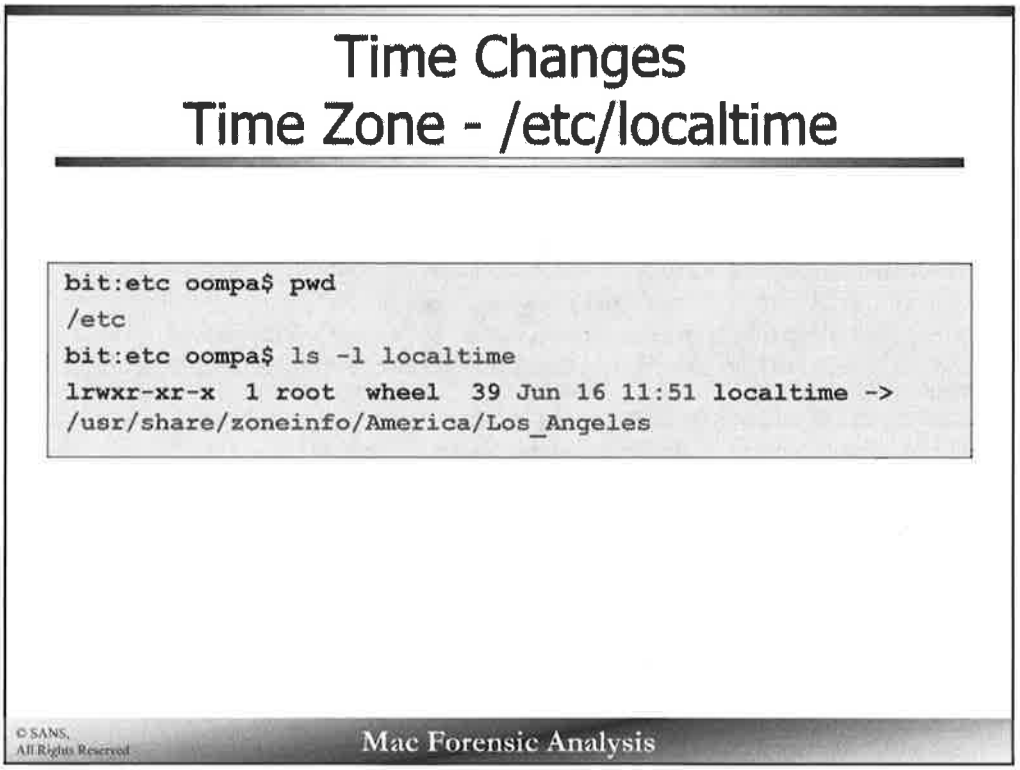
A user may make intentional changes to the system time or date. They might, for example, be traveling and need to change time zones. A user may also make changes to the system time or date to throw a temporal investigation off.



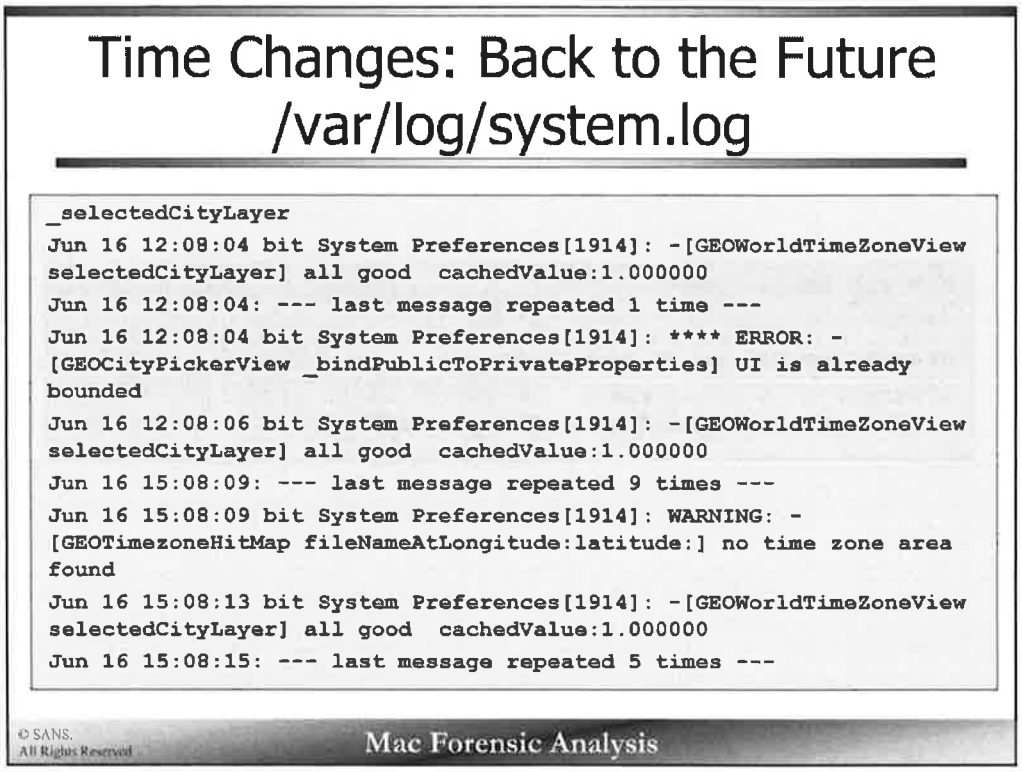
A user might attempt to hide or obfuscate activity by changing the system time.

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.

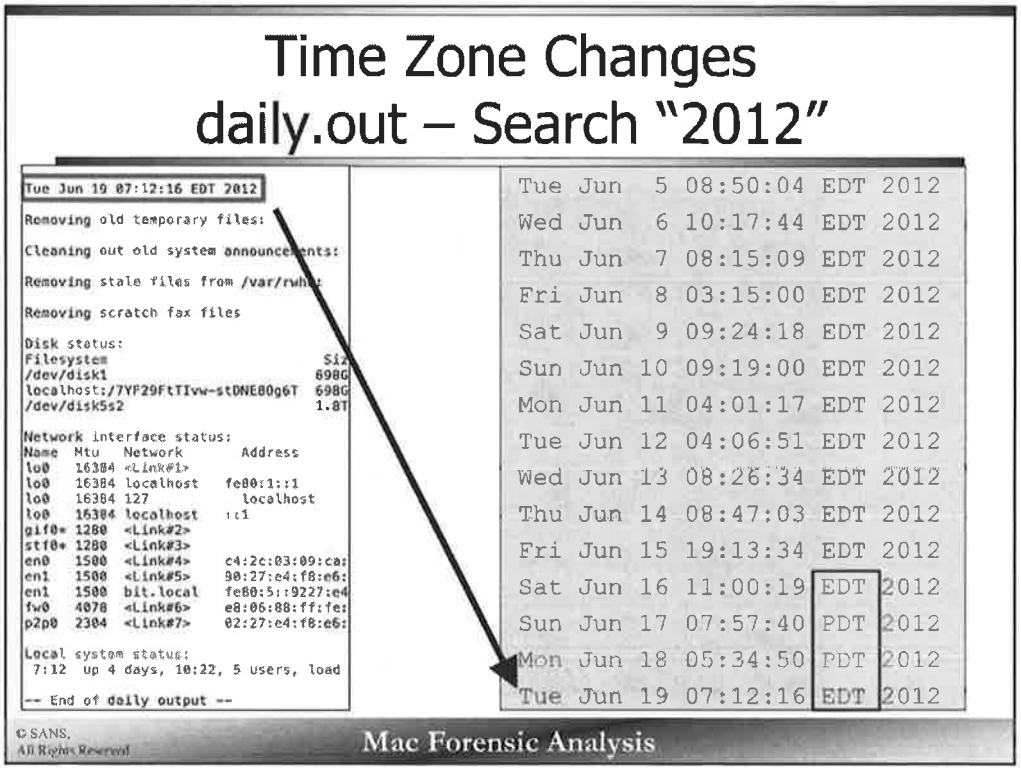
```
Jun 16 14:50:56 bit System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] Invalidating _selectedCityLayer
Jun 16 14:50:56 bit System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] all good  cachedValue:1.000000
Jun 16 14:50:56: --- last message repeated 4 times ---
Jun 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 System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] Invalidating _selectedCityLayer
Jun 16 11:51:06 bit System Preferences[1828]: -[GEOWorldTimeZoneView
selectedCityLayer] all good  cachedValue:1.000000
Jun 16 11:51:06 bit ntpd[1848]: proto: precision = 1.000 usec
```



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 16th.

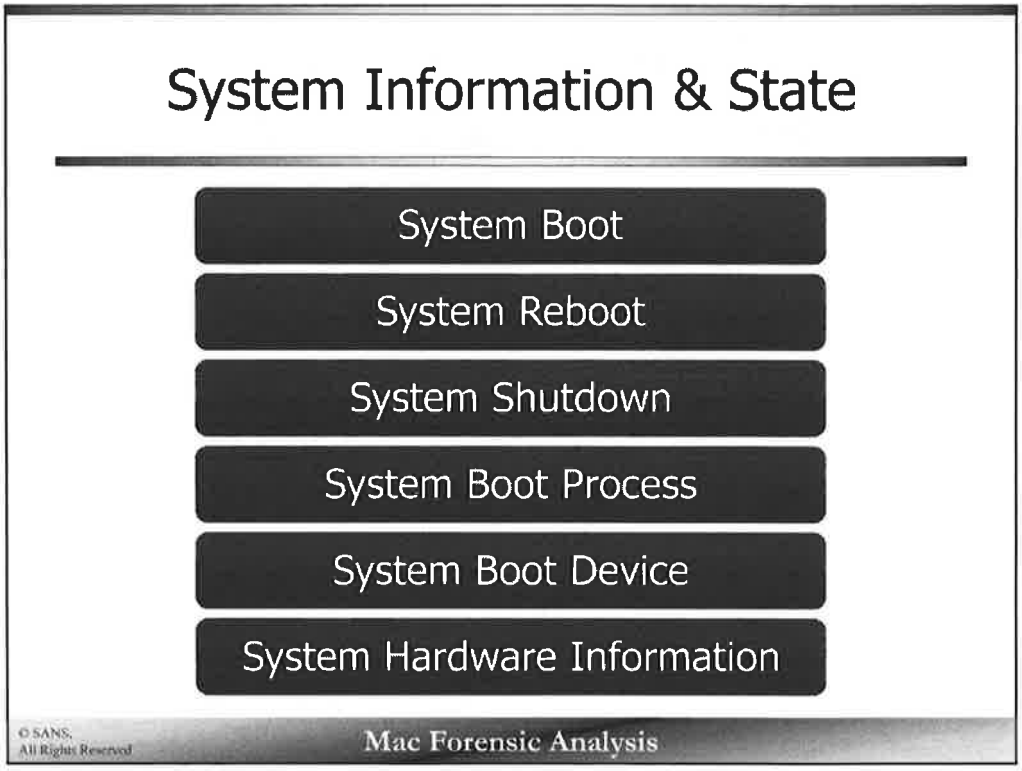


The time can also be changed to the "future" as shown above. The time zone was changed to reflect three hours in advance. The `system.log` will show entries one right after another with the difference in time., note the same relative minute and second values.



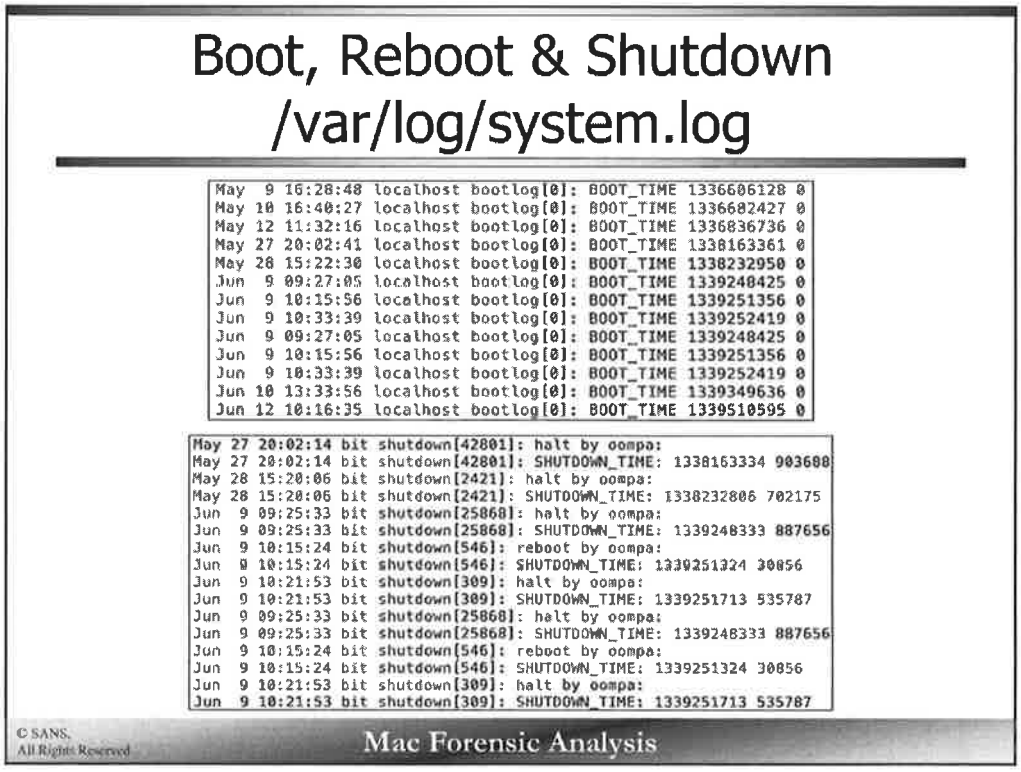
Time zone changes can also be shown in the `daily.out` file.

In the example above a search, was completed for the year "`2012`". The highlighted entries show a change from Eastern Daylight Time (EDT) to Pacific Daylight Time (PDT) for the days of June 17th and June 18th.



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.

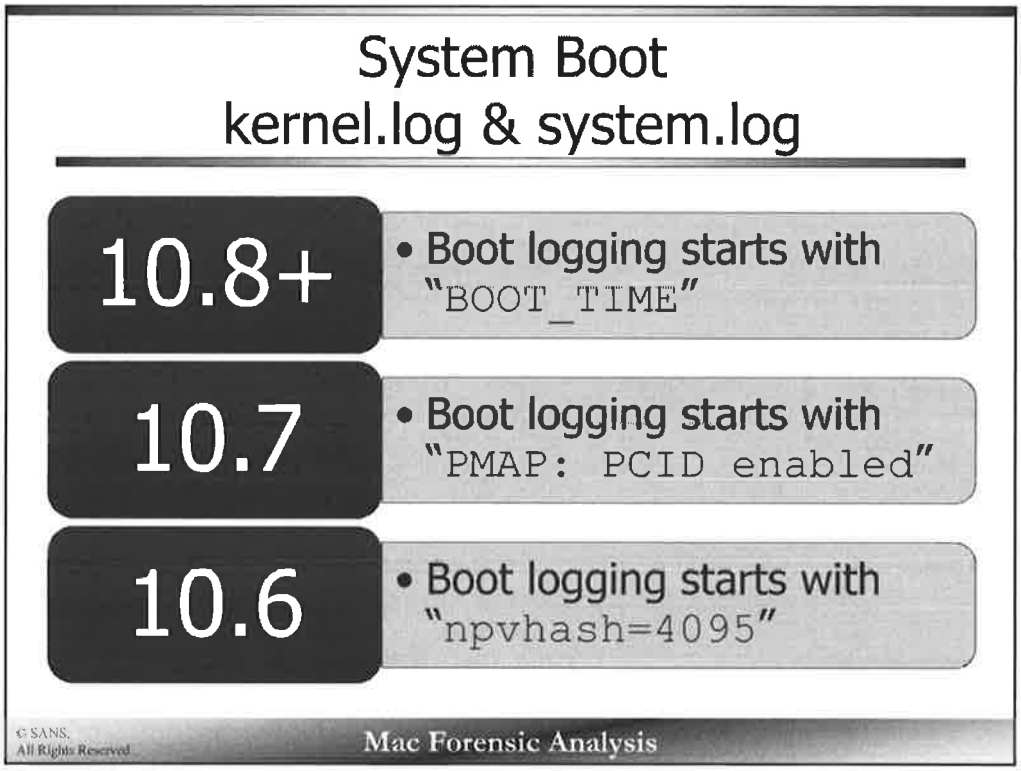
The hardware configuration of the system can be determined to see if the system was upgraded or if the boot drive changed systems.



Knowing when the system boots or when it is shutdown can help with the timeline aspect of some investigations. The `system.log` contains the terms "`BOOT_TIME`" and "`SHUTDOWN_TIME`". The log also records what user account shutdown ("`halt`") or restarted ("`reboot`") the system.

```
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
```

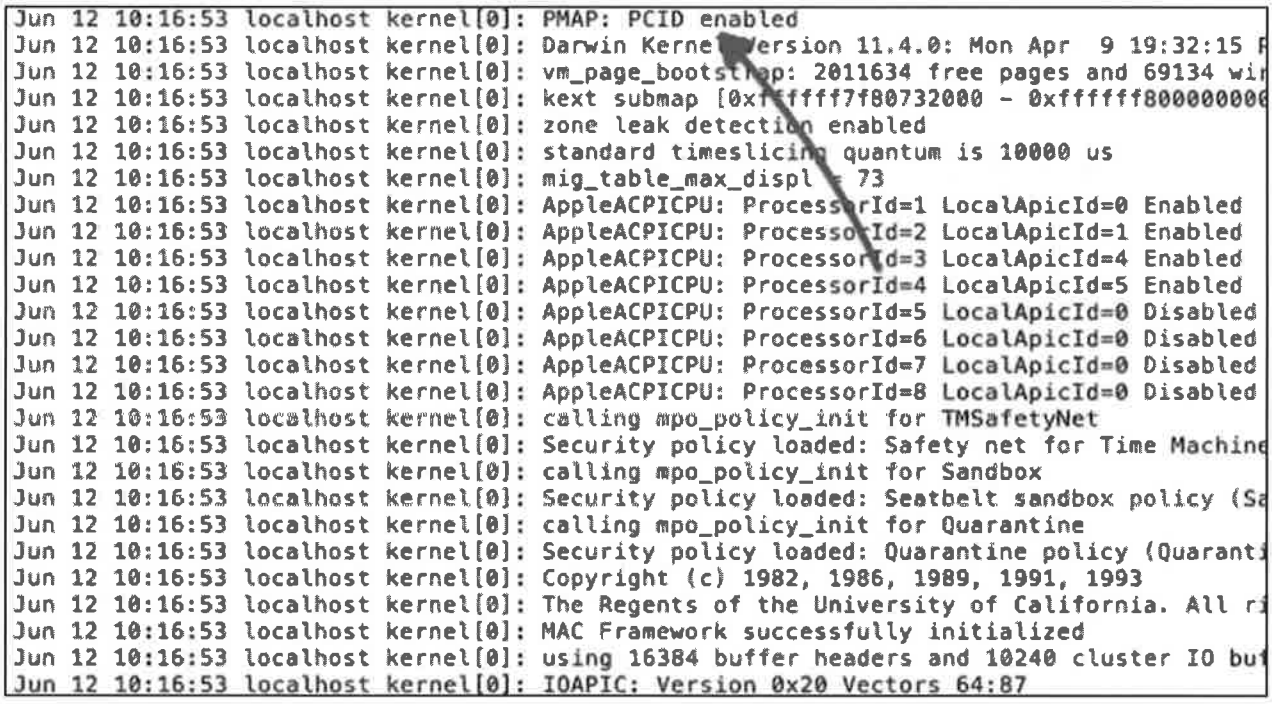
```
May 27 20:02:14 bit shutdown[42801]: halt by oompa:
May 27 20:02:14 bit shutdown[42801]: SHUTDOWN_TIME: 1338163334 903688
May 28 15:20:06 bit shutdown[2421]: halt by oompa:
May 28 15:20:06 bit shutdown[2421]: SHUTDOWN_TIME: 1338232806 702175
Jun  9 09:25:33 bit shutdown[25868]: halt by oompa:
Jun  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
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
```

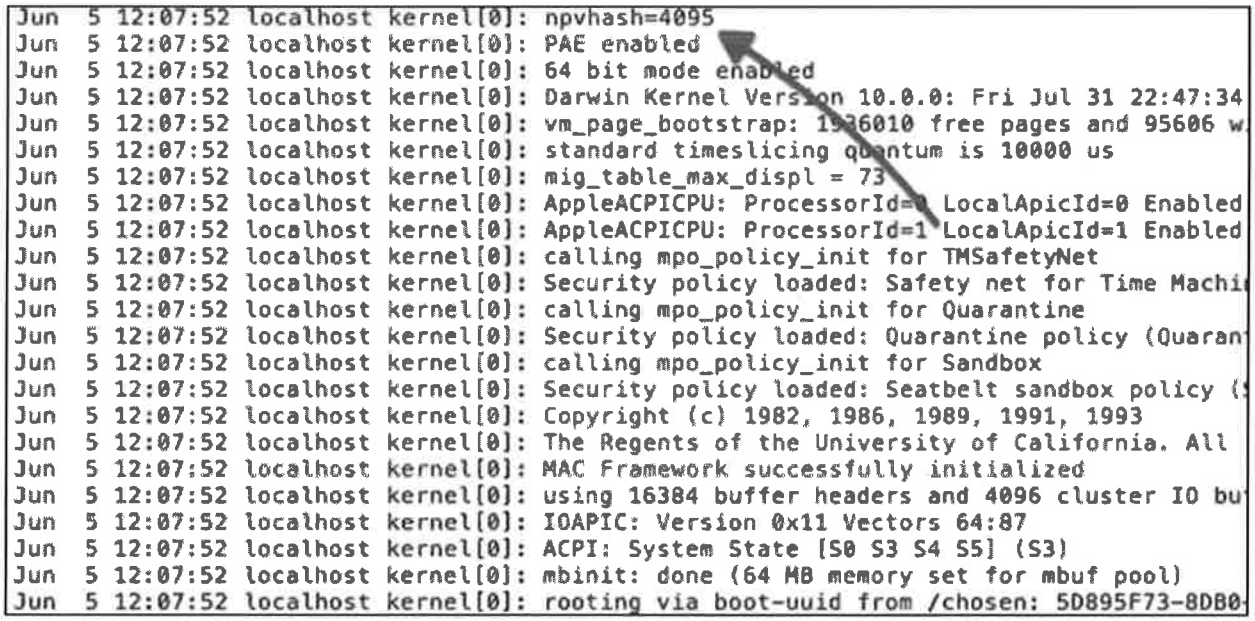


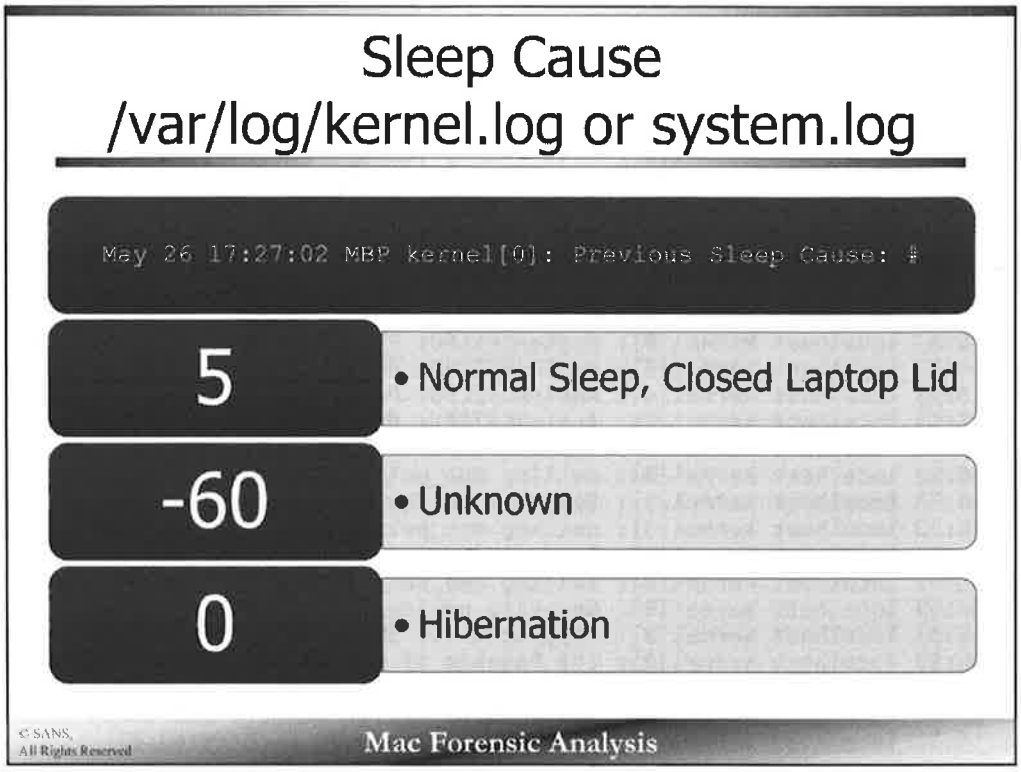
In the `kernel.log` or `system.log` (depending on the system), you will find the example text shown above when boot logging starts.

Due to the log compilation of all the `system.log` in 10.8, the boot logging starts with `BOOT_TIME`, so it is relatively easy to recognize. In 10.7 the boot logging starts with the line "`PMAP: PCID enabled`" in the `kernel.log`. 10.6 starts with the entry "`npvhash=4095`"

The screenshot examples shown above are the 10.7 (top) and 10.6 (bottom) versions of the `kernel.log`. 10.8+ systems will show the same as the top screenshot, however the boot logging is clearly shown with the "`BOOT_TIME`" entry.

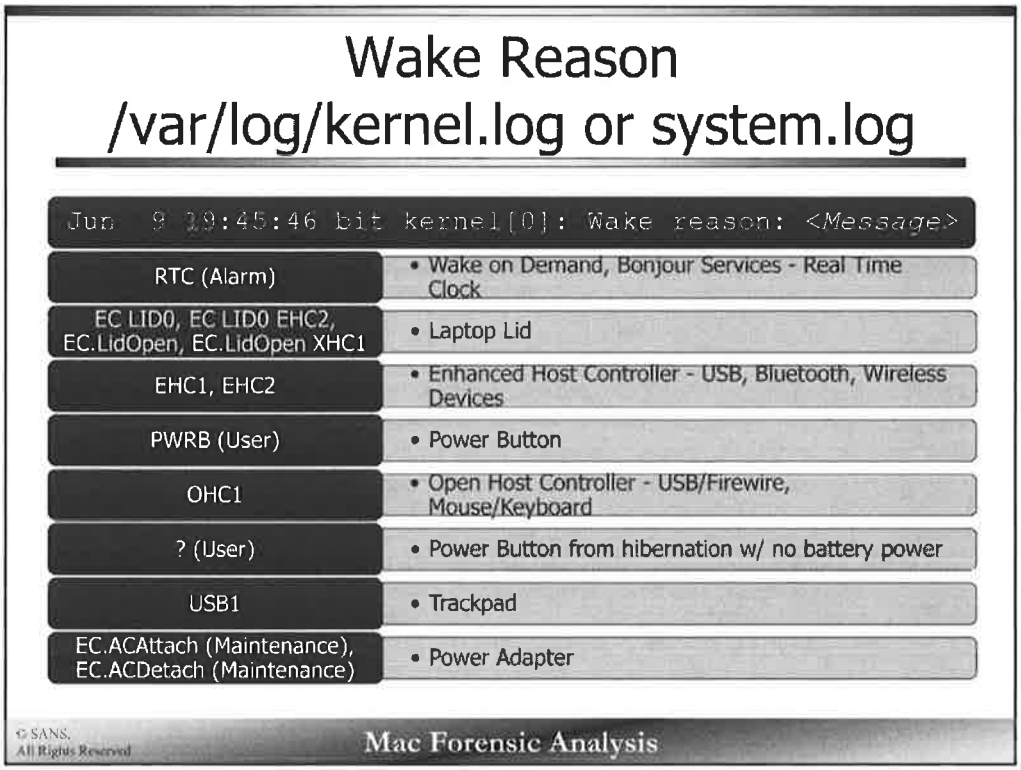

Jun 12 10:16:53 localhost kernel[0]: PMAP: PCID enabled
Jun 12 10:16:53 localhost kernel[0]: Darwin Kernel Version 11.4.0: Mon Apr 9 19:32:15
Jun 12 10:16:53 localhost kernel[0]: vm_page_bootstrap: 2011634 free pages and 69134 wi
Jun 12 10:16:53 localhost kernel[0]: kext submap [0xffffff7f80732000 - 0xffffff8000000
Jun 12 10:16:53 localhost kernel[0]: zone leak detection enabled
Jun 12 10:16:53 localhost kernel[0]: standard timeslicing quantum is 10000 us
Jun 12 10:16:53 localhost kernel[0]: mig_table_max_displ = 73
Jun 12 10:16:53 localhost kernel[0]: AppleACPICPU: ProcessorId=1 LocalApicId=0 Enabled
Jun 12 10:16:53 localhost kernel[0]: AppleACPICPU: ProcessorId=2 LocalApicId=1 Enabled
Jun 12 10:16:53 localhost kernel[0]: AppleACPICPU: ProcessorId=3 LocalApicId=4 Enabled
Jun 12 10:16:53 localhost kernel[0]: AppleACPICPU: ProcessorId=4 LocalApicId=5 Enabled
Jun 12 10:16:53 localhost kernel[0]: AppleACPICPU: ProcessorId=5 LocalApicId=0 Disabled
Jun 12 10:16:53 localhost kernel[0]: AppleACPICPU: ProcessorId=6 LocalApicId=0 Disabled
Jun 12 10:16:53 localhost kernel[0]: AppleACPICPU: ProcessorId=7 LocalApicId=0 Disabled
Jun 12 10:16:53 localhost kernel[0]: AppleACPICPU: ProcessorId=8 LocalApicId=0 Disabled
Jun 12 10:16:53 localhost kernel[0]: calling mpo_policy_init for TMSafetyNet
Jun 12 10:16:53 localhost kernel[0]: Security policy loaded: Safety net for Time Machine
Jun 12 10:16:53 localhost kernel[0]: calling mpo_policy_init for Sandbox
Jun 12 10:16:53 localhost kernel[0]: Security policy loaded: Seatbelt sandbox policy (Sa
Jun 12 10:16:53 localhost kernel[0]: calling mpo_policy_init for Quarantine
Jun 12 10:16:53 localhost kernel[0]: Security policy loaded: Quarantine policy (Quarantin
Jun 12 10:16:53 localhost kernel[0]: Copyright (c) 1982, 1986, 1989, 1991, 1993
Jun 12 10:16:53 localhost kernel[0]: The Regents of the University of California. All ri
Jun 12 10:16:53 localhost kernel[0]: MAC Framework successfully initialized
Jun 12 10:16:53 localhost kernel[0]: using 16384 buffer headers and 10240 cluster IO buf
Jun 12 10:16:53 localhost kernel[0]: IOAPIC: Version 0x20 Vectors 64:87


Jun 5 12:07:52 localhost kernel[0]: npvhash=4095
Jun 5 12:07:52 localhost kernel[0]: PAE enabled
Jun 5 12:07:52 localhost kernel[0]: 64 bit mode enabled
Jun 5 12:07:52 localhost kernel[0]: Darwin Kernel Version 10.0.0: Fri Jul 31 22:47:34
Jun 5 12:07:52 localhost kernel[0]: vm_page_bootstrap: 1936010 free pages and 95606 w
Jun 5 12:07:52 localhost kernel[0]: standard timeslicing quantum is 10000 us
Jun 5 12:07:52 localhost kernel[0]: mig_table_max_displ = 73
Jun 5 12:07:52 localhost kernel[0]: AppleACPICPU: ProcessorId=0 LocalApicId=0 Enabled
Jun 5 12:07:52 localhost kernel[0]: AppleACPICPU: ProcessorId=1 LocalApicId=1 Enabled
Jun 5 12:07:52 localhost kernel[0]: calling mpo_policy_init for TMSafetyNet
Jun 5 12:07:52 localhost kernel[0]: Security policy loaded: Safety net for Time Machi
Jun 5 12:07:52 localhost kernel[0]: calling mpo_policy_init for Quarantine
Jun 5 12:07:52 localhost kernel[0]: Security policy loaded: Quarantine policy (Quaran
Jun 5 12:07:52 localhost kernel[0]: calling mpo_policy_init for Sandbox
Jun 5 12:07:52 localhost kernel[0]: Security policy loaded: Seatbelt sandbox policy (
Jun 5 12:07:52 localhost kernel[0]: Copyright (c) 1982, 1986, 1989, 1991, 1993
Jun 5 12:07:52 localhost kernel[0]: The Regents of the University of California. All
Jun 5 12:07:52 localhost kernel[0]: MAC Framework successfully initialized
Jun 5 12:07:52 localhost kernel[0]: using 16384 buffer headers and 4096 cluster IO bu
Jun 5 12:07:52 localhost kernel[0]: IOAPIC: Version 0x11 Vectors 64:87
Jun 5 12:07:52 localhost kernel[0]: ACPI: System State [S0 S3 S4 S5] (S3)
Jun 5 12:07:52 localhost kernel[0]: mbinit: done (64 MB memory set for mbuf pool)
Jun 5 12:07:52 localhost kernel[0]: rooting via boot-uuid from /chosen: 5D895F73-8DB0-

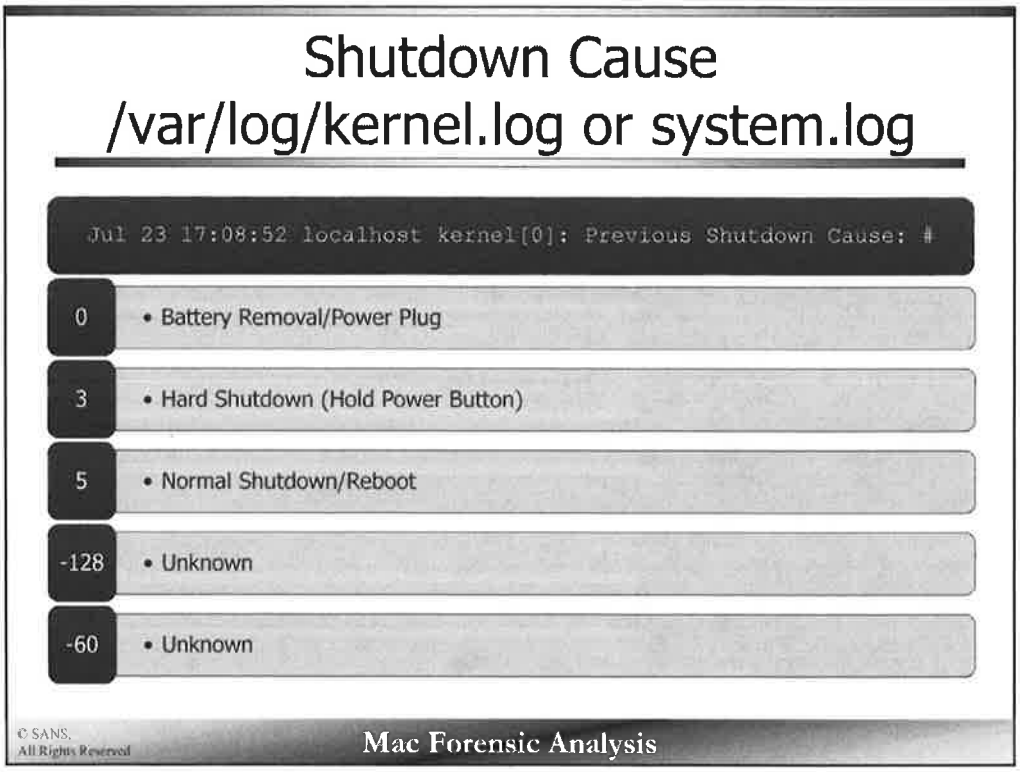


The system records reasons the computer has been put to sleep, which is known as "`Sleep Cause`". Outlined in the screenshot above are some of the reasons have been documented.

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



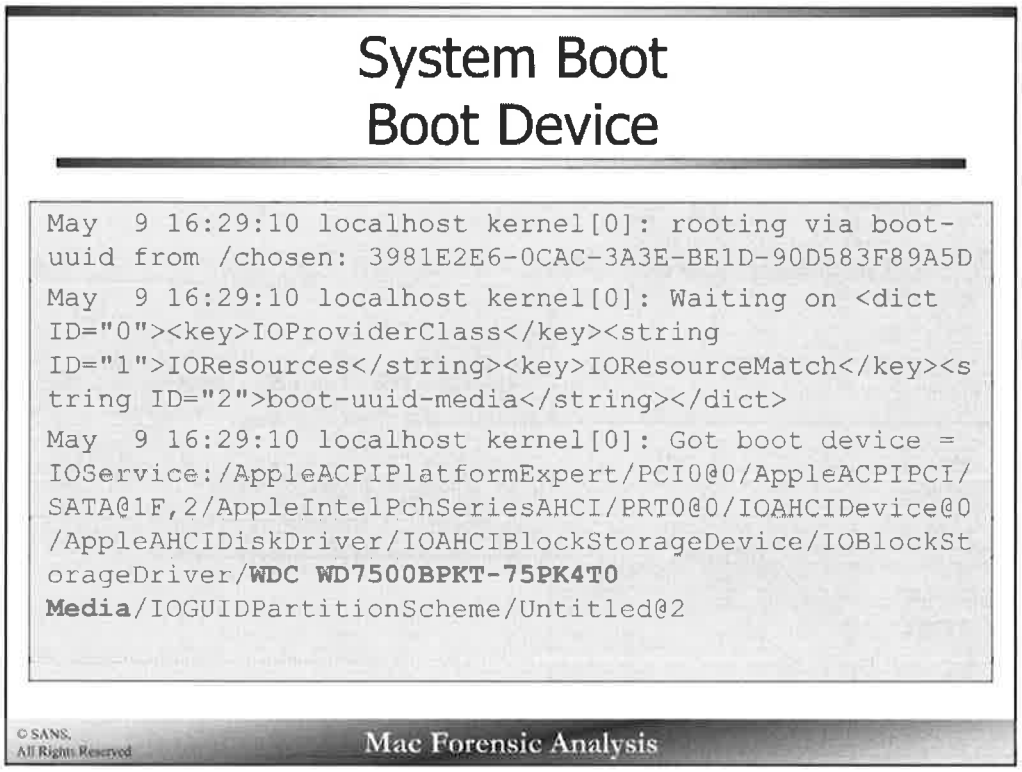
The system records the reasons the computer has awoken, known as "`Wake Reason`". Outlined in the screenshot above are some of the reasons.



The system records the reasons the computer was shut down, known as "`Shutdown Cause`". Outlined in the screenshot above are some of the reasons.

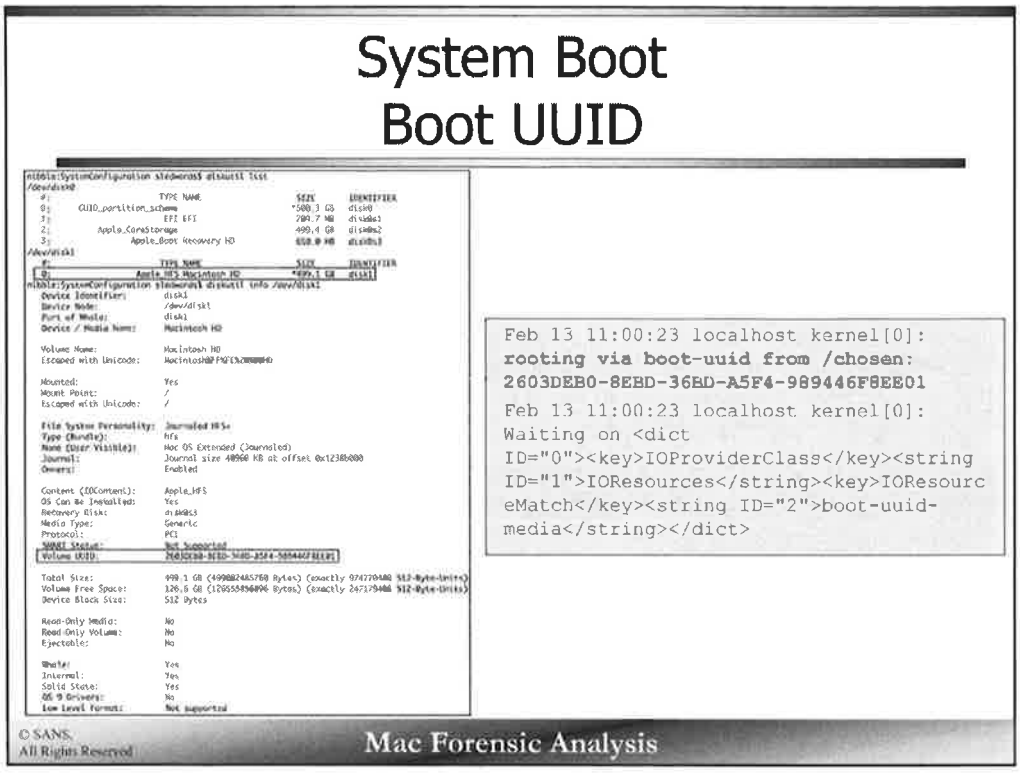
The "`Shutdown Cause`" that should be most prevalent on a working system should be cause number "`5`".

The documentation on these Sleep/Wake/Shutdown reasons are limited and more experimentation and research need to be performed.



The system boot device is always listed in the boot logging which is shown in either the `kernel.log` or the `system.log` (depending on the OS X version). The UUID highlighted above is the UUID of the boot volume, which in most cases is labeled as the "`Macintosh HD`" volume.

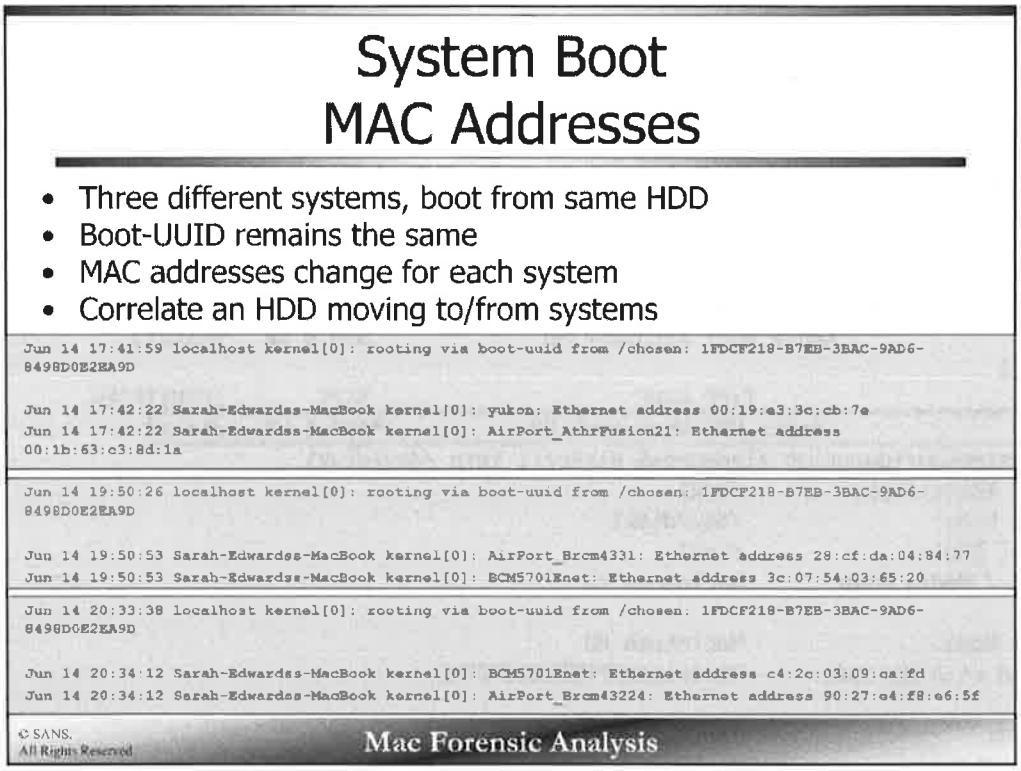
The other highlighted string shows some information about the boot device. In the example shown we can tell that the hard drive on the system (and with a quick Google search) is a Western Digital 750GB hard drive.



Some Mac systems such as the older Mac Pro, can have hard drives easily removed and changed. The boot UUID is unique to that volume on that specific hard drive. The UUID will not change if the hard drive is removed and put into a different system. We can search for the term "`/chosen`" to see if the system was booted from a different volume or if the hard drive was used to boot multiple systems.

The author once had an instance where they needed to find out what systems a particular hard drive was placed into (and consequently booted from). It turns out the same hard drive was booted from three different systems throughout the years, and it was easy to correlate it using the `kernel.log` (or `system.log`). The author was able to determine, using MAC addresses (next slide), that this volume was booted from three different systems. The boot UUID did not change, but the MAC addresses did change – indicating separate machines. The author was also able to physically locate each system by their MAC addresses, for further analysis.

```
nibble:SystemConfiguration sledwards$ diskutil list
/dev/disk0
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *500.3 GB   disk0
   1:                        EFI EFI                     209.7 MB   disk0s1
   2:          Apple_CoreStorage                        499.4 GB   disk0s2
   3:                 Apple_Boot Recovery HD            650.0 MB   disk0s3
/dev/disk1
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:                 Apple_HFS Macintosh HD           *499.1 GB   disk1
nibble:SystemConfiguration sledwards$ diskutil info /dev/disk1
   Device Identifier:        disk1
   Device Node:              /dev/disk1
   Part of Whole:            disk1
   Device / Media Name:      Macintosh HD

   Volume Name:              Macintosh HD
   Escaped with Unicode:     Macintosh%FF%FE%20%00HD

   Mounted:                  Yes
   Mount Point:              /
   Escaped with Unicode:     /

   File System Personality:  Journaled HFS+
   Type (Bundle):            hfs
   Name (User Visible):      Mac OS Extended (Journaled)
   Journal:                  Journal size 40960 KB at offset 0x1238b000
   Owners:                   Enabled

   Content (IOContent):      Apple_HFS
   OS Can Be Installed:      Yes
   Recovery Disk:            disk0s3
   Media Type:               Generic
   Protocol:                 PCI
   SMART Status:             Not Supported
   Volume UUID:              2603DEB0-8EBD-36BD-A5F4-989446F8EE01

   Total Size:               499.1 GB (499082485760 Bytes) (exactly 974770480 512-Byte-Units)
   Volume Free Space:        126.6 GB (126555856896 Bytes) (exactly 247179408 512-Byte-Units)
   Device Block Size:        512 Bytes

   Read-Only Media:          No
   Read-Only Volume:         No
   Ejectable:                No

   Whole:                    Yes
   Internal:                 Yes
   Solid State:              Yes
   OS 9 Drivers:             No
   Low Level Format:         Not supported
```



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). Right off the bat 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_Bcrm4331` – Broadcom network card
  - `BCM5701Enet` – Broadcom network card
- System 3
  - `BCM5701Enet` – Broadcom network card
  - `Airport_Bcrm43224` – 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

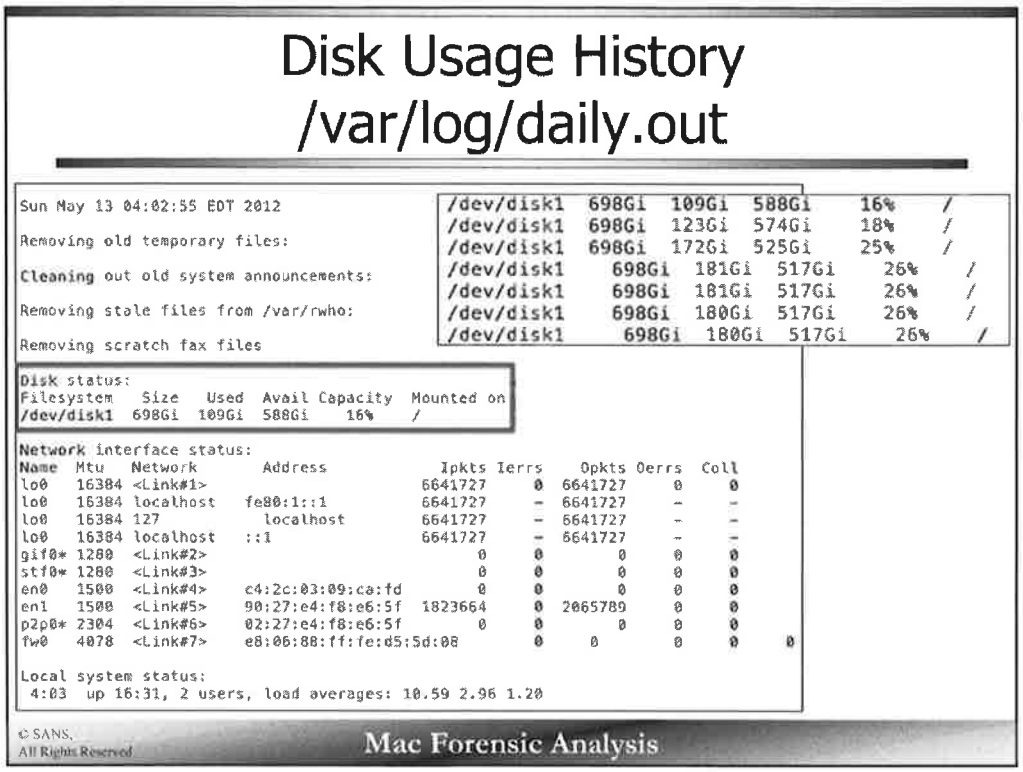
Jun 14 19:50:53 Sarah-Edwardss-MacBook kernel[0]: AirPort_Brcm4331: Ethernet address 28:cf:da:04:84:77

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

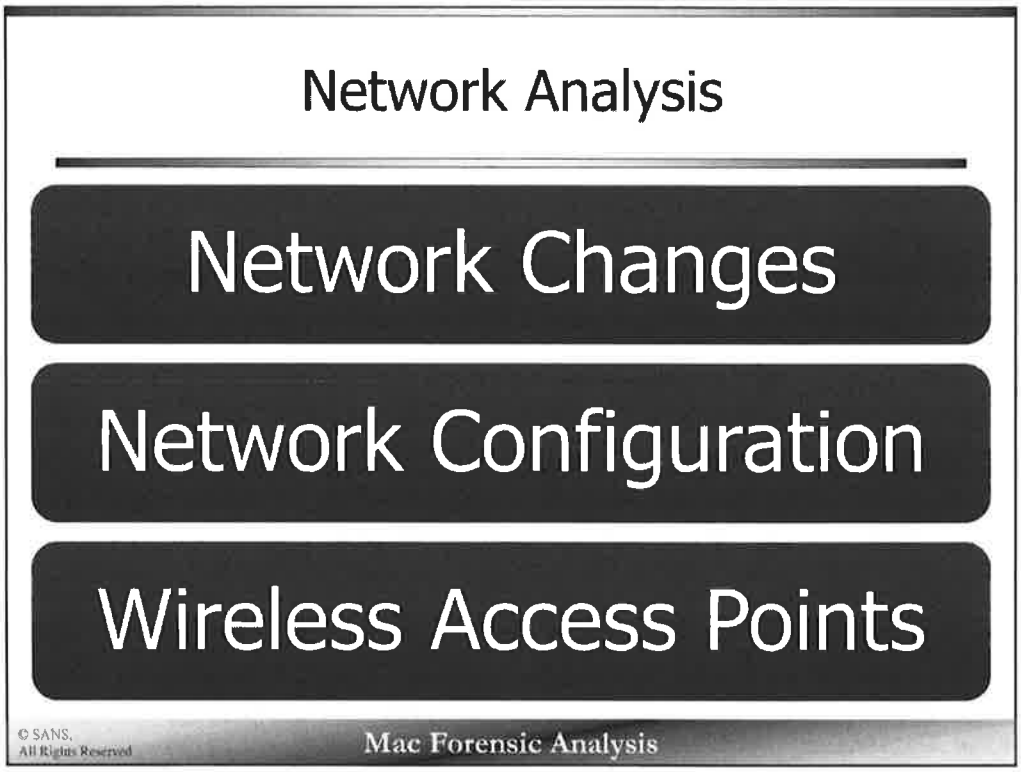
Jun 14 20:34:12 Sarah-Edwardss-MacBook kernel[0]: AirPort_Brcm43224: Ethernet address 90:27:e4:f8:e6:5f



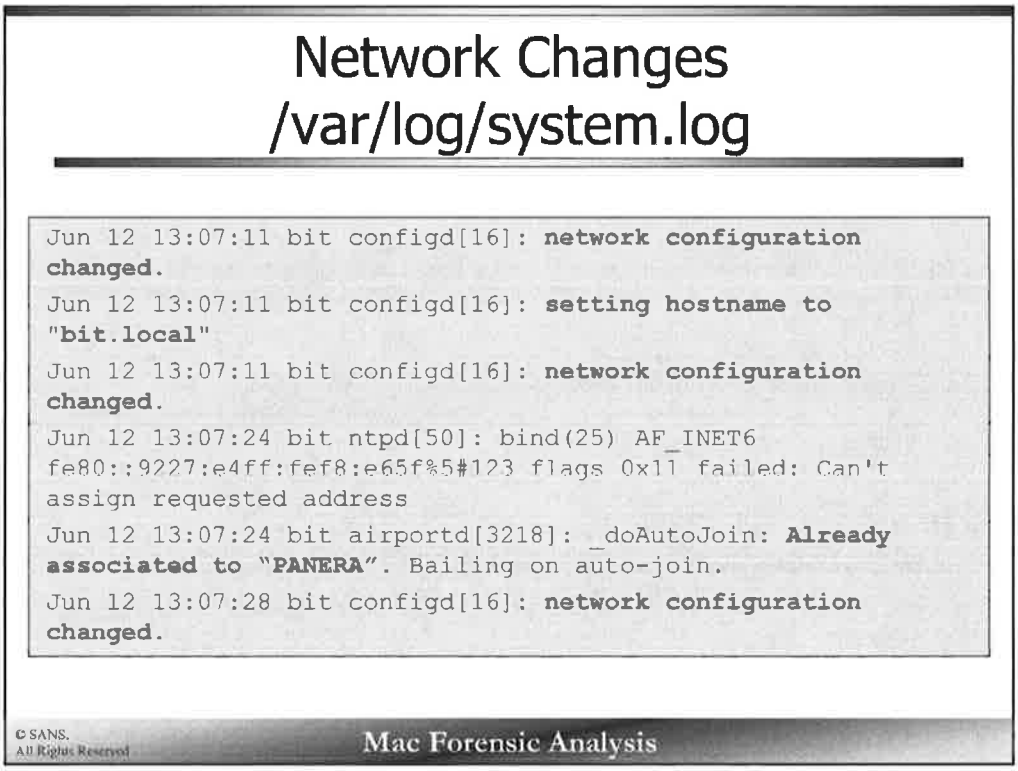
The disk usage for a system might be useful to see how much disk space a system uses over time. The `daily.out` is one of the log files associated with the Unix maintenance scripts (along with `weekly.out` and `monthly.out`), which records "`Disk Status`". We can "`grep`" this log file for the specific disk (`/dev/disk1`) as shown in the example above. The example shows that disk space out has increased from 16% to 26%.

```
Sun May 13 04:02:55 EDT 2012

Removing old temporary files:

Cleaning out old system announcements:

Removing stale files from /var/rwho:

Removing scratch fax files

Disk status:
Filesystem    Size   Used  Avail Capacity  Mounted on
/dev/disk1   698Gi  109Gi  588Gi    16%    /

Network interface status:
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16384 <Link#1>                      6641727     0  6641727     0     0
lo0   16384 localhost   fe80:1::1         6641727     -  6641727     -     -
lo0   16384 127           localhost       6641727     -  6641727     -     -
lo0   16384 localhost   ::1               6641727     -  6641727     -     -
gif0* 1280  <Link#2>                            0     0        0     0     0
stf0* 1280  <Link#3>                            0     0        0     0     0
en0   1500  <Link#4>    c4:2c:03:09:ca:fd       0     0        0     0     0
en1   1500  <Link#5>    90:27:e4:f8:e6:5f 1823664     0  2065789     0     0
p2p0* 2304  <Link#6>    02:27:e4:f8:e6:5f       0     0        0     0     0
fw0   4078  <Link#7>    e8:06:88:ff:fe:d5:5d:08       0     0        0     0     0

Local system status:
 4:03  up 16:31, 2 users, load averages: 10.59 2.96 1.20
```

```
/dev/disk1   698Gi   109Gi   588Gi     16%     /
/dev/disk1   698Gi   123Gi   574Gi     18%     /
/dev/disk1   698Gi   172Gi   525Gi     25%     /
/dev/disk1    698Gi   181Gi   517Gi     26%     /
/dev/disk1    698Gi   181Gi   517Gi     26%     /
/dev/disk1    698Gi   180Gi   517Gi     26%     /
/dev/disk1     698Gi   180Gi   517Gi     26%     /
```



The network access of a system can help investigators determine what type of access a system had at a particular time. Travel can be shown by viewing access to various wireless access points.

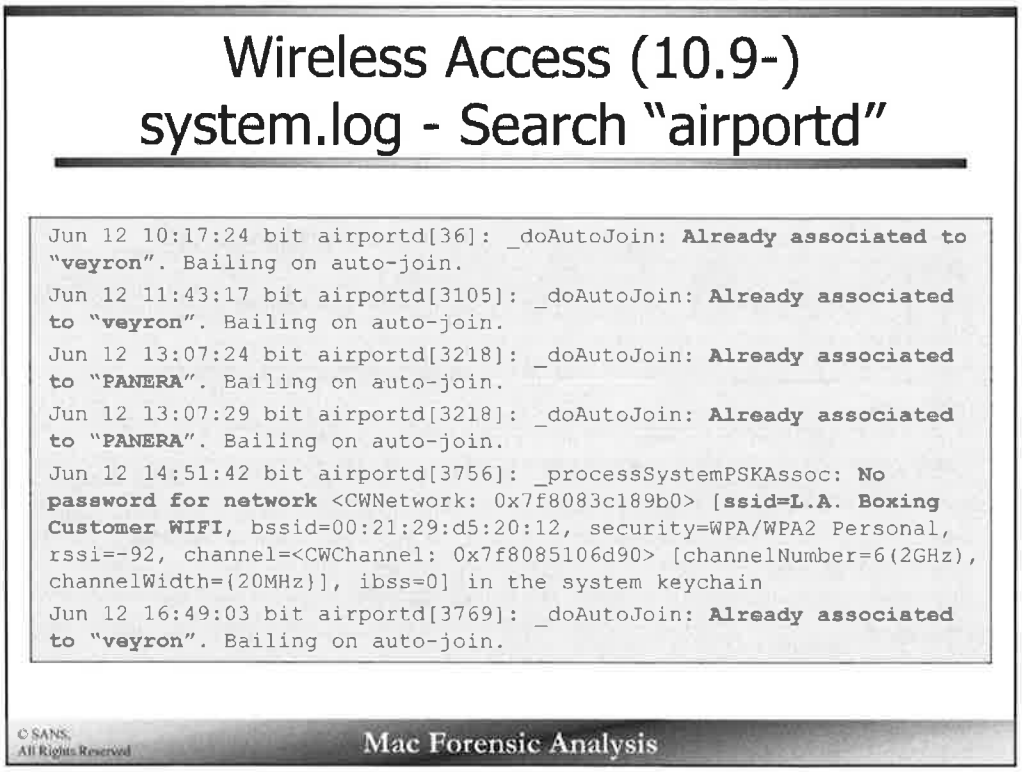


Searches for keywords such as "`configd`", "`airportd`" and "`ntpd`" will show network configuration changes.

In the screenshot above the local system changed the hostname to "`bit.local`" and shows that the system was already connected to an access point named "`PANERA`".

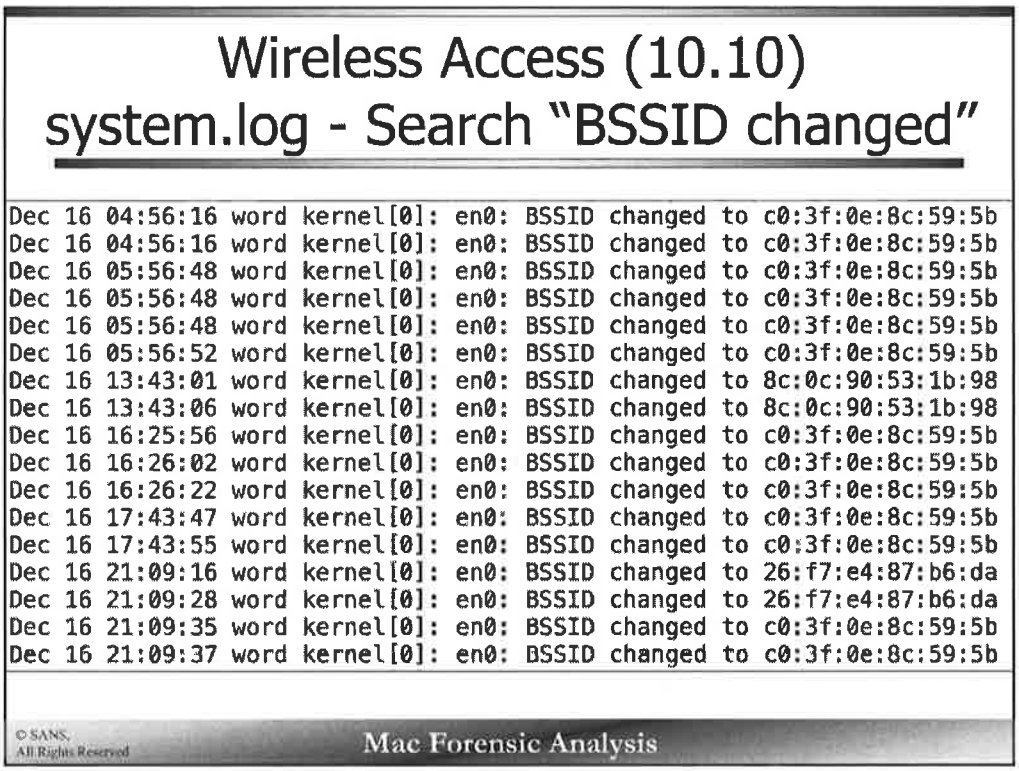
Most desktop systems may not have as many "network configuration changes" as they may have static network settings. Laptops, due to their mobility, tend to change more often to connect to different access points.

These times are stored in local system time.



The `system.log` can be searched for "`airportd`" to determine when a system had been associated with a specific wireless access point (`veyron`, `PANERA`). It may also show if an attempt was made to associate to a wireless access point (`L.A. Boxing Customer WIFI`, shown in the screenshot above).

These times are stored in local system time.

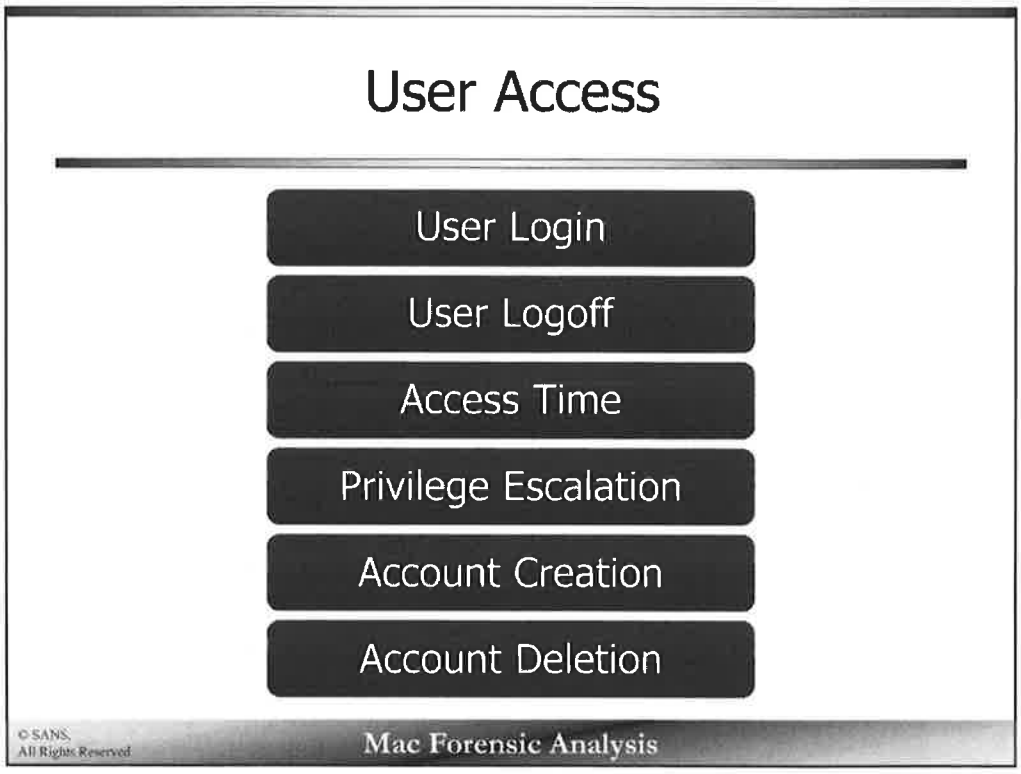


The wireless access configuration logs changed in 10.10. They no long provide SSID names, instead they record the MAC address of the access point.

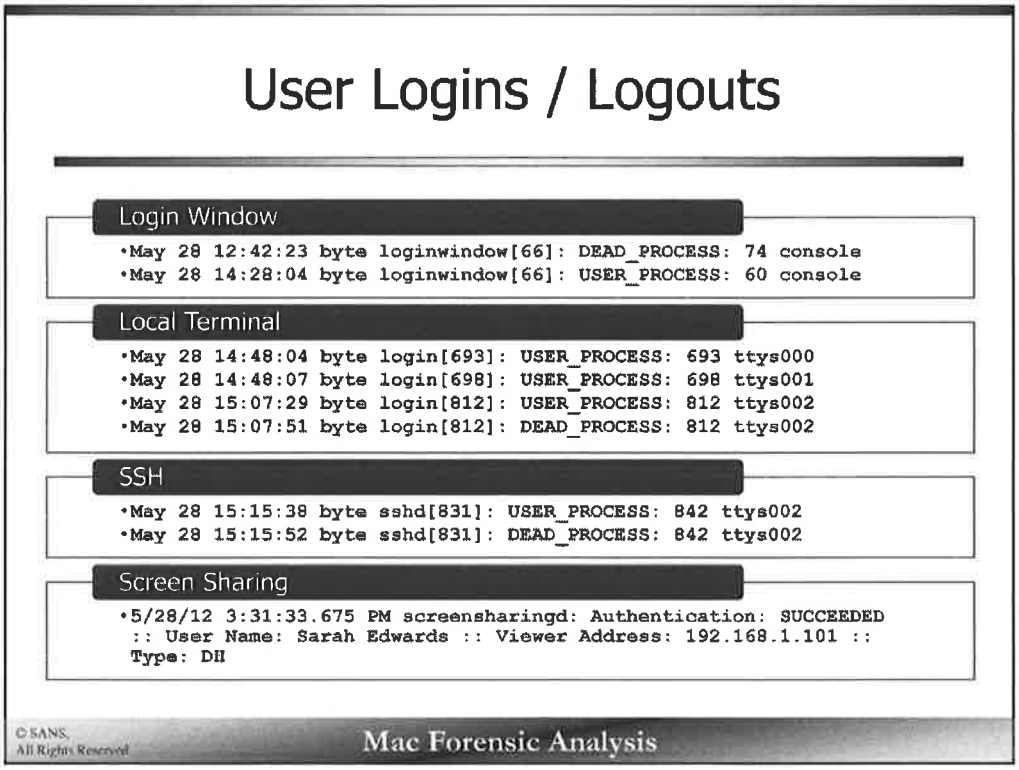
In the example above, the system used three BSSIDs:

- `c0:3f:0e:8c:59:5b`
- `8c:0c:90:53:1b:98`
- `26:f7:e4:87:b6:da`

You may be able to correlate these MAC addresses with names using the `com.apple.airport.preferences.plist` file.



Logs can tell a lot about how each user makes use of a system. When did they login or logoff, do they have privileged access, and how long were they 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.

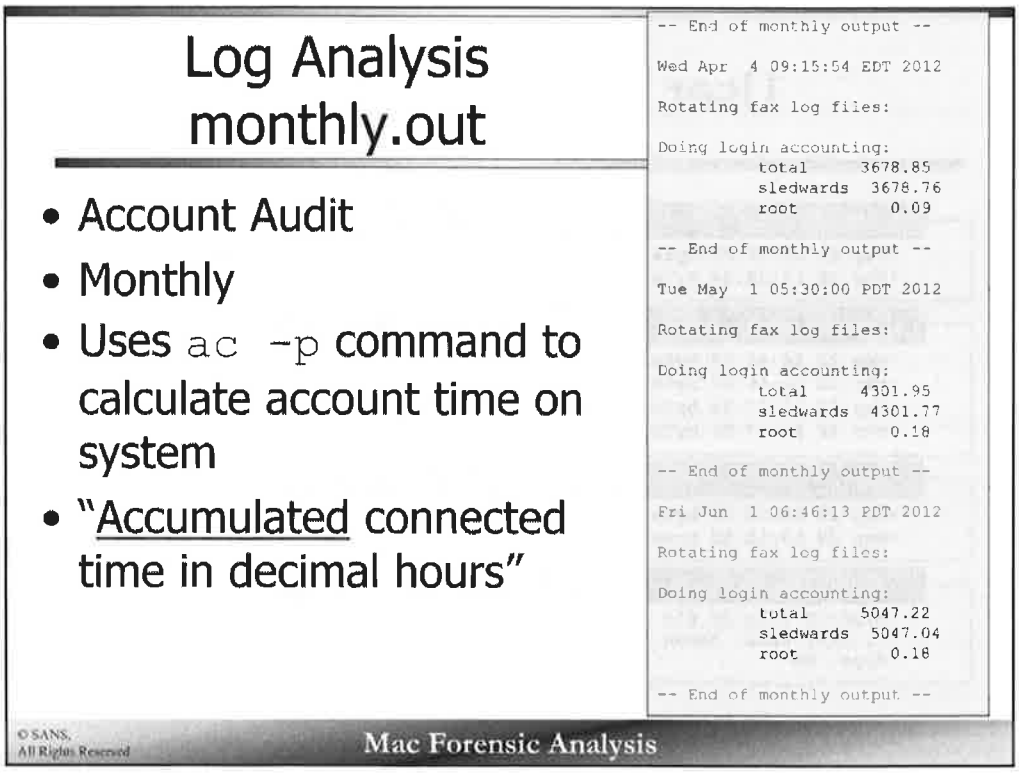
In the Local Terminal example, we can see the results of three logins (PIDs 693, 698, 812) but only one logoff (PID 812). Three terminal windows have been opened, ttys000, ttys001 and ttys002. The terminal window labeled ttys002 has been exited out of or closed. Note this type uses the '`login`' process.

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.

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.

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.

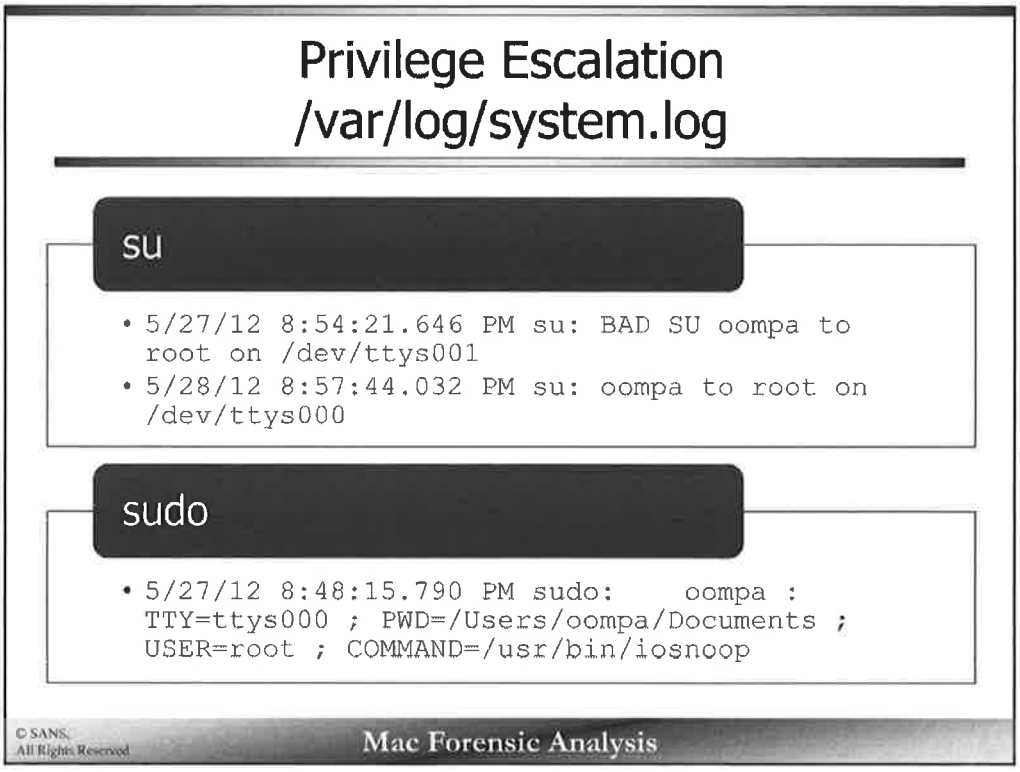
These login/logout messages can be found in the `system.log` or the Apple System Logs. These events in the ASL logs contain more detailed information such as which user logged in.



The monthly maintenance script output, shown in the screenshot, contains data about user accounts on the system. This can be used to determine which accounts get used more often than others.

The time next to each account name is the accumulated connection time, in decimal hours, created by running the '`ac -p`' command.

The example shows the `sledwards` account gets used much more often than the other root account. It is worth noting that the root account is not enabled by default on OS X systems. The output shows the root user account being used slightly more in May than it was in April.

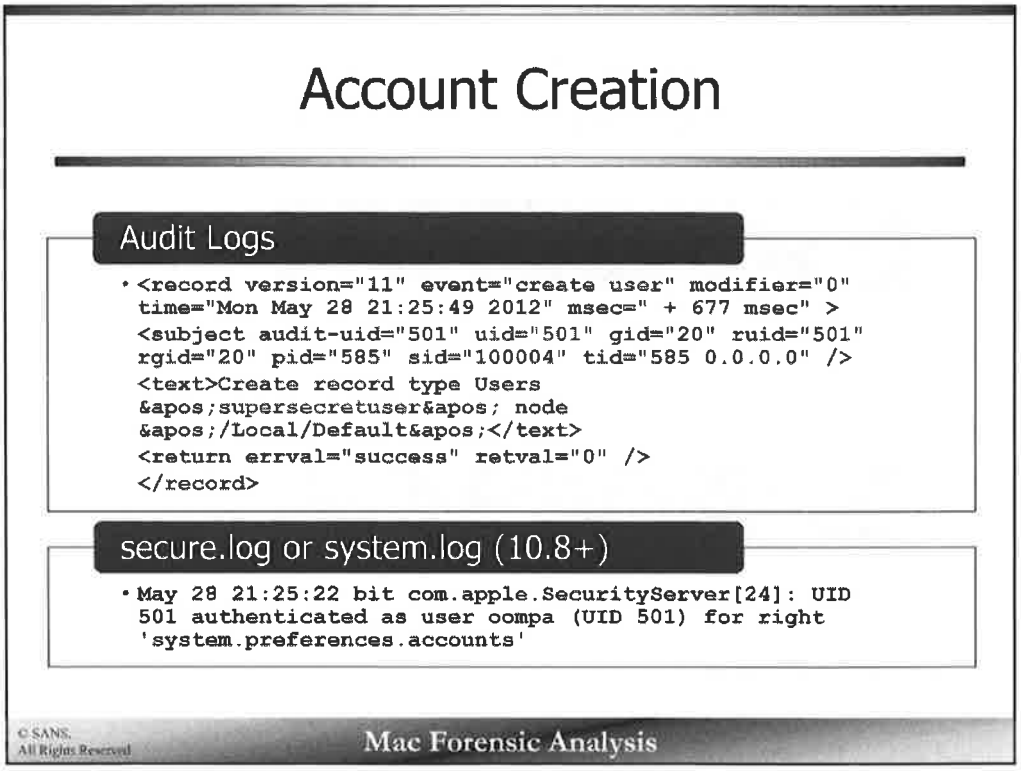


Privilege escalation is needed when a Standard user needs to perform root level actions. The command `su` can be used to login as root (or another user) for a session, while the `sudo` command will run a command as root for five minutes (this time can be changed in the `/etc/sudoers` configuration file).

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



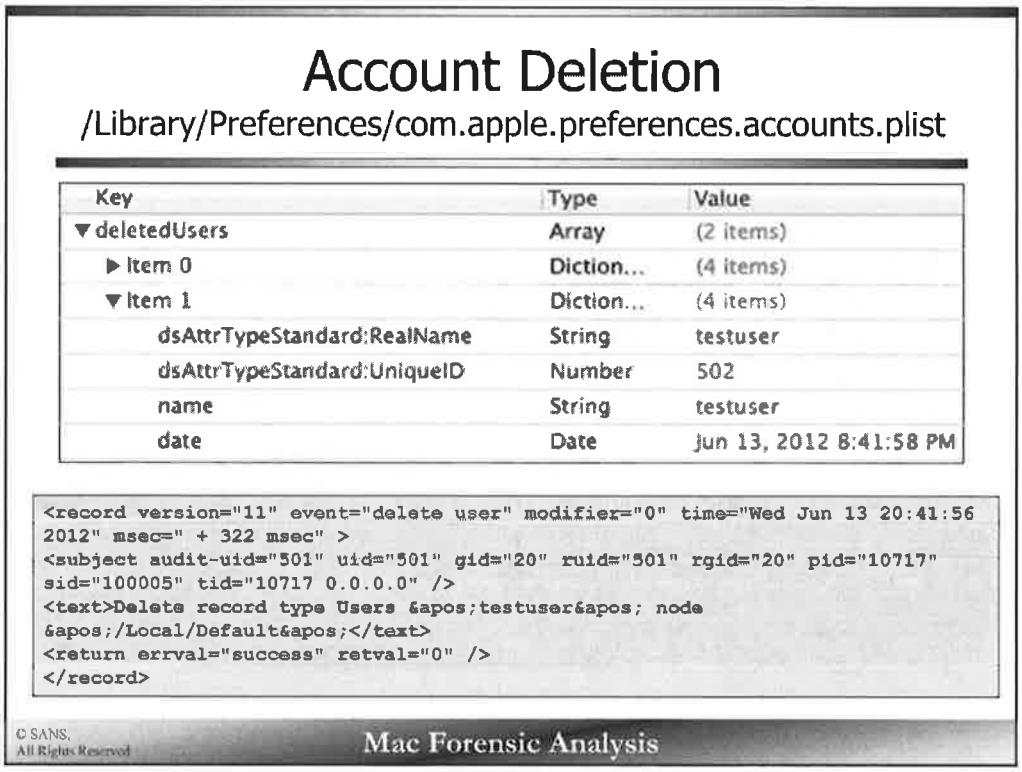
User account creation can also be of immense interest to an investigator. Audit records for user creation are very verbose. Lots of related entries are created. Audit records also have more detail than records found in the `secure.log` (or `system.log`).

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 `secure.log` (or the `system.log` with 10.8+) 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.

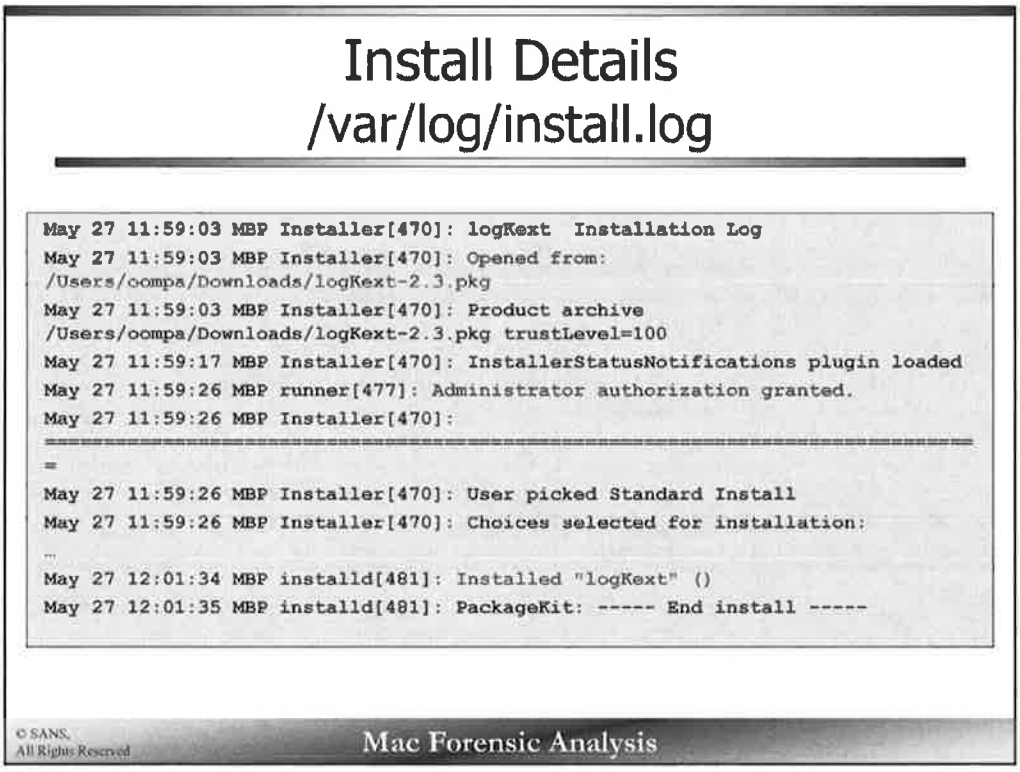
This is a good example of using multiple sources to get a more detailed picture.



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

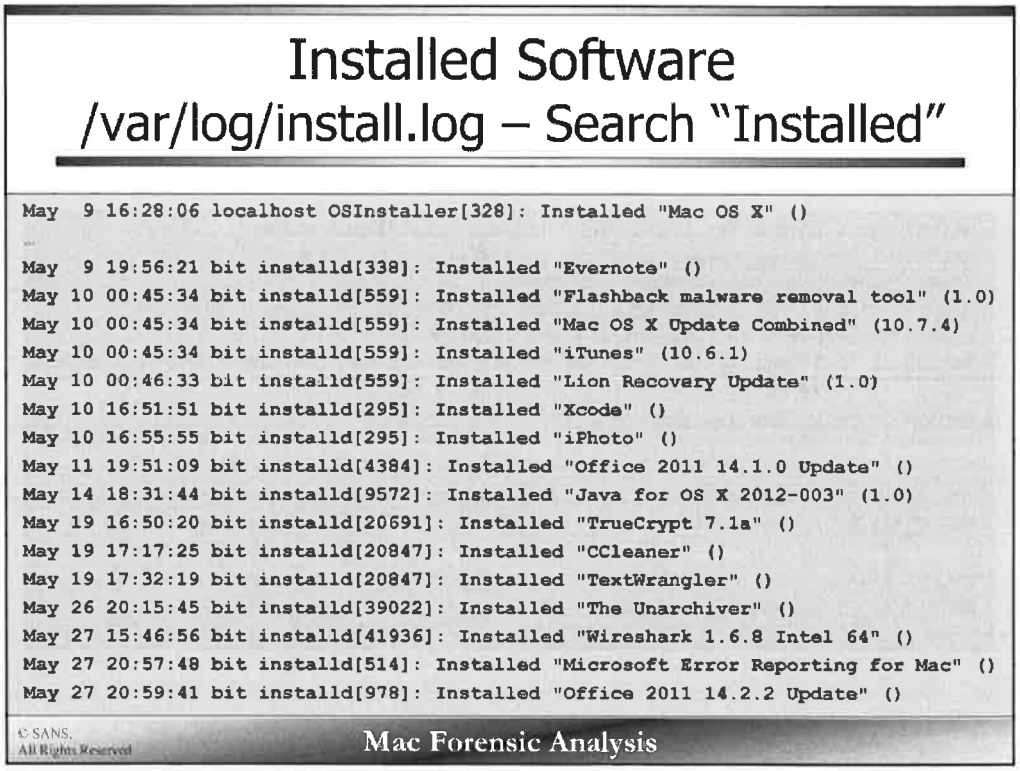
The deleted users are shown in the `com.apple.preferences.accounts.plist` file under the `deletedUsers` key. This key contains the deleted user's name, UID, username, and the deletion date in local system time.

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.



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

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

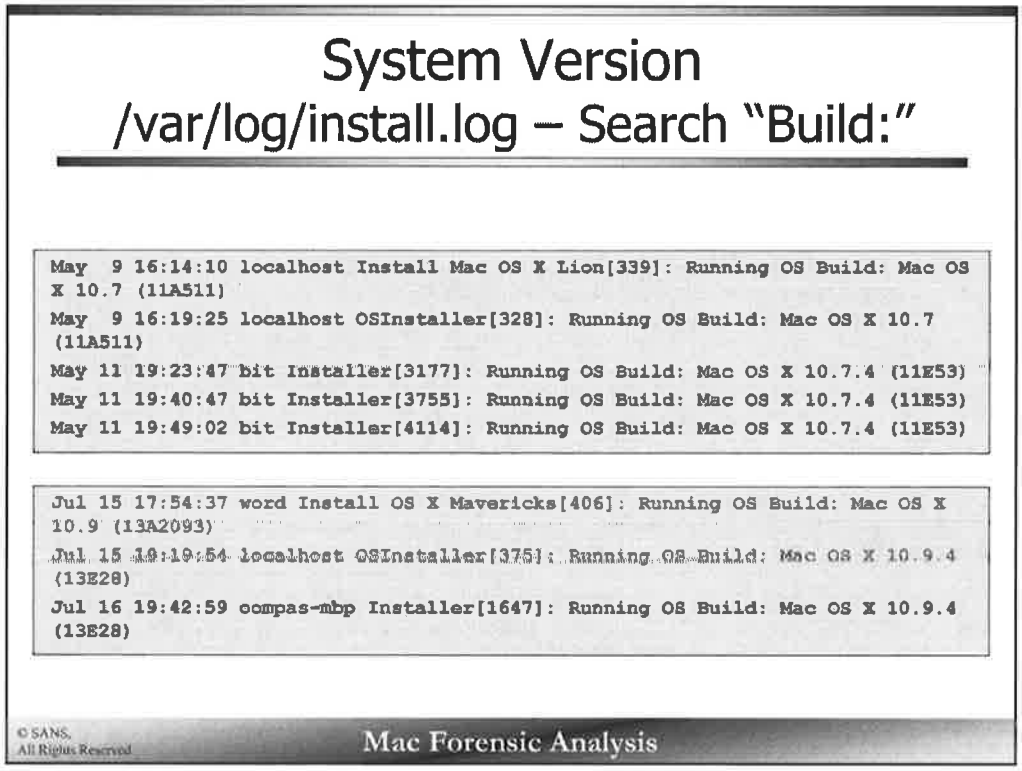


Software installations can be a good resource to find out what types of software are used on the system and when they were downloaded.

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.

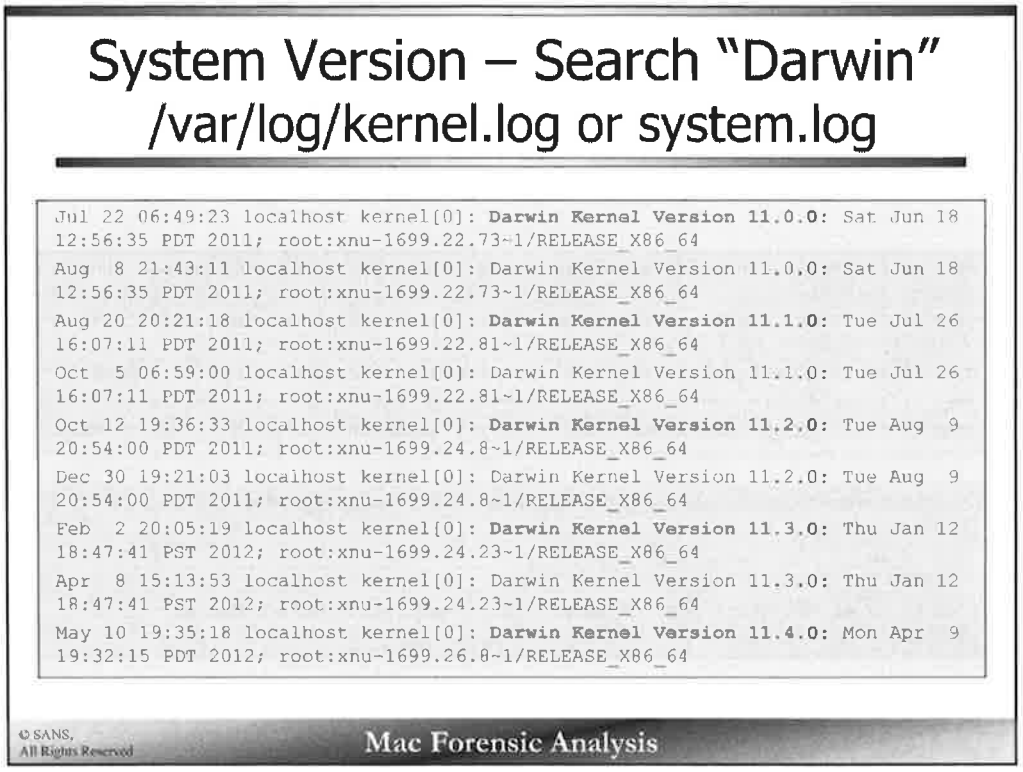
Caveat: Software installed via a "Drag and Drop" method such as Firefox and Chrome will not show in this log. Some software allows a user to copy the application directly to the `/Application` directory.



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

In the first example above, the system was first installed on May 9th with Mac OS X 10.7. A small clue to tell us this is the first install is that the hostname of the system has not yet been set.

The second example shows the same artifacts for a 10.9 to 10.9.4 system install.



Using the kernel log (10.7 and below) or the `system.log` (10.8+), we can view the kernel versions in use at a given time by searching for the term "`Darwin`". The kernel versions can be traced back to specific OS X versions.

For example:

Kernel version 11.0.0 is OS X version 10.7

Kernel version 11.4.0 is OS X version 10.7.4

Kernel version 12.0.0 is OS X version 10.8

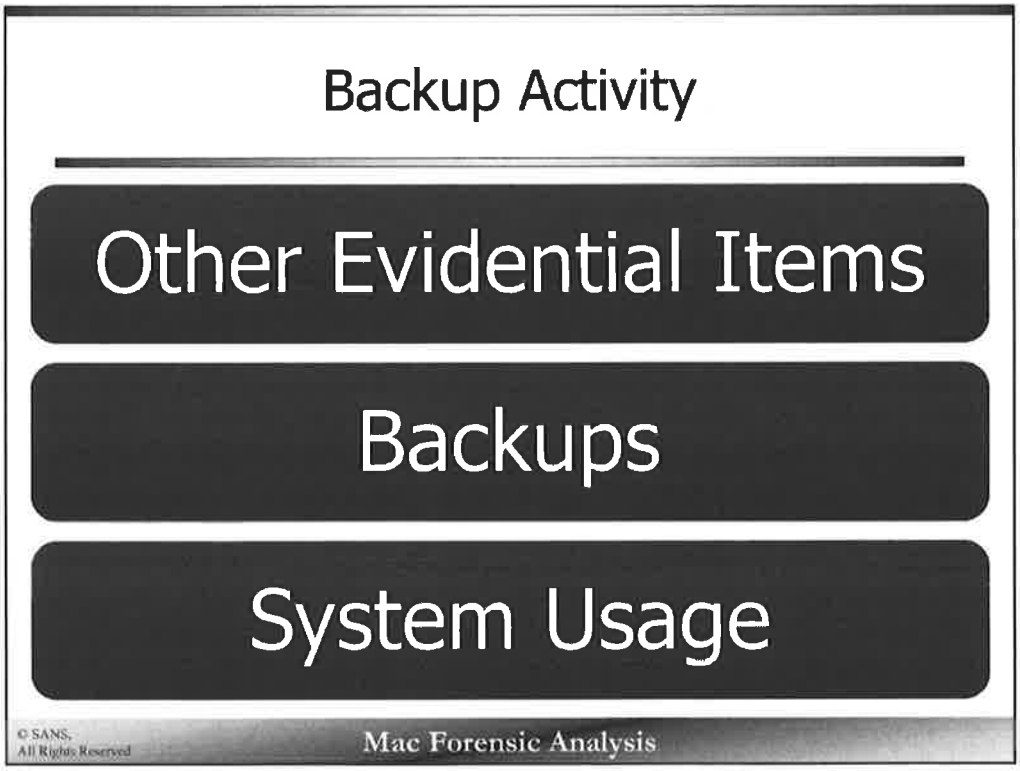
Kernel version 12.5.0 is OS X version 10.8.5

Kernel version 13.0.0 is OS X version 10.9

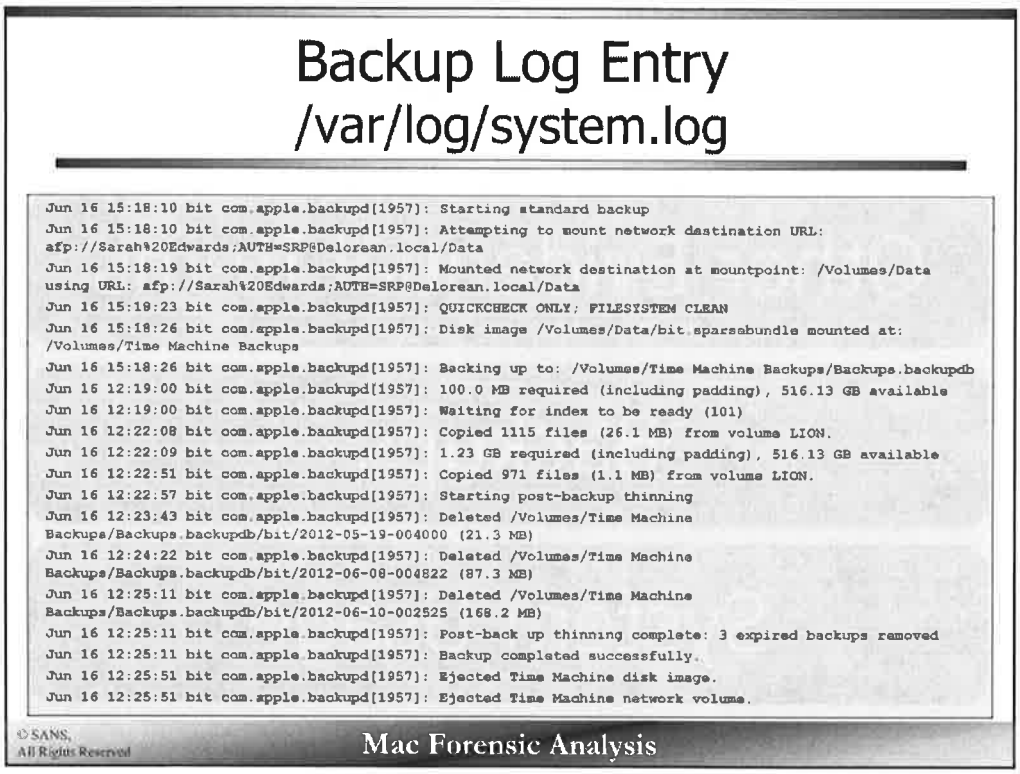
Kernel version 14.0.0 is OS X version 10.10

Reference:

http://en.wikipedia.org/wiki/Darwin_(operating_system)



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 and 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 completed

Jun 16 15:18:10 bit com.apple.backupd[1957]: **Starting standard backup**

Jun 16 15:18:10 bit com.apple.backupd[1957]: Attempting to mount network destination URL: **afp://Sarah%20Edwards;AUTH=SRP@Delorean.local/Data**

Jun 16 15:18:19 bit com.apple.backupd[1957]: Mounted network destination at mountpoint: /Volumes/Data using URL: afp://Sarah%20Edwards;AUTH=SRP@Delorean.local/Data

Jun 16 15:18:23 bit com.apple.backupd[1957]: QUICKCHECK ONLY; FILESYSTEM CLEAN

Jun 16 15:18:26 bit com.apple.backupd[1957]: Disk image **/Volumes/Data/bit.sparsebundle** mounted at: **/Volumes/Time Machine Backups**

Jun 16 15:18:26 bit com.apple.backupd[1957]: Backing up to: /Volumes/Time Machine Backups/Backups.backupdb

Jun 16 12:19:00 bit com.apple.backupd[1957]: 100.0 MB required (including padding), 516.13 GB available

Jun 16 12:19:00 bit com.apple.backupd[1957]: Waiting for index to be ready (101)

Jun 16 12:22:08 bit com.apple.backupd[1957]: Copied 1115 files (26.1 MB) from volume **LION.**

Jun 16 12:22:09 bit com.apple.backupd[1957]: 1.23 GB required (including padding), 516.13 GB available

Jun 16 12:22:51 bit com.apple.backupd[1957]: Copied 971 files (1.1 MB) from volume LION.

Jun 16 12:22:57 bit com.apple.backupd[1957]: Starting post-backup thinning

Jun 16 12:23:43 bit com.apple.backupd[1957]: **Deleted /Volumes/Time Machine Backups/Backups.backupdb/bit/2012-05-19-004000 (21.3 MB)**

Jun 16 12:24:22 bit com.apple.backupd[1957]: Deleted /Volumes/Time Machine Backups/Backups.backupdb/bit/2012-06-08-004822 (87.3 MB)

Jun 16 12:25:11 bit com.apple.backupd[1957]: Deleted /Volumes/Time Machine Backups/Backups.backupdb/bit/2012-06-10-002525 (168.2 MB)

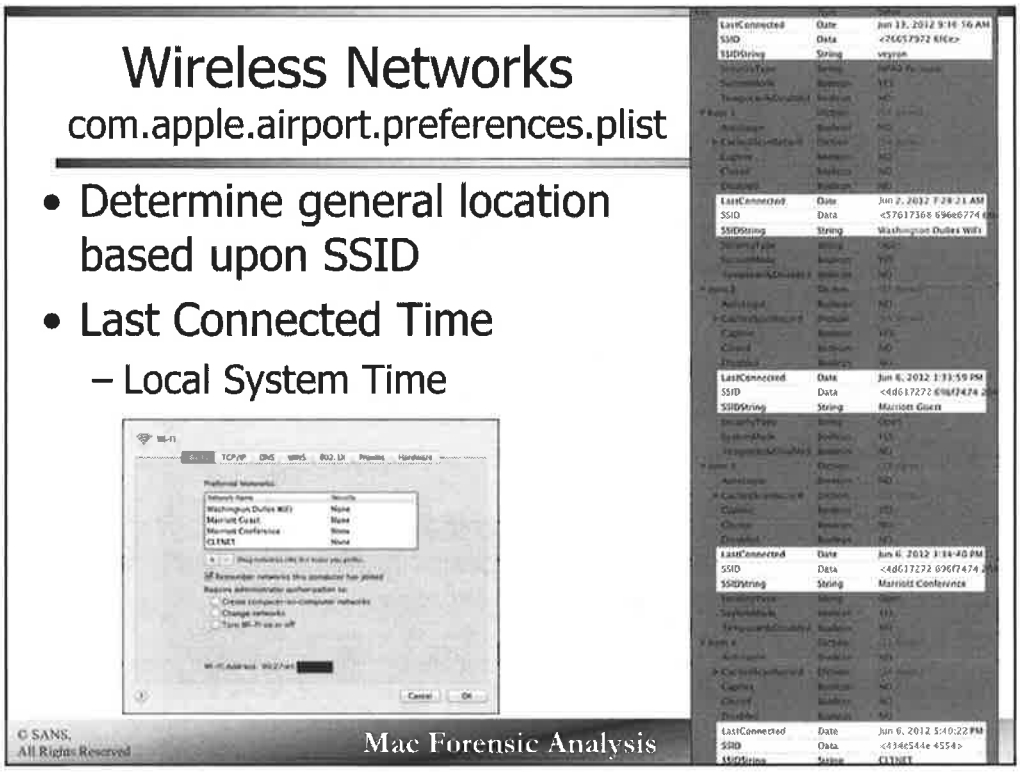
Jun 16 12:25:11 bit com.apple.backupd[1957]: Post-back up thinning complete: 3 expired backups removed

Jun 16 12:25:11 bit com.apple.backupd[1957]: **Backup completed successfully.**

Jun 16 12:25:51 bit com.apple.backupd[1957]: Ejected Time Machine disk image.

Jun 16 12:25:51 bit com.apple.backupd[1957]: Ejected Time Machine network volume.

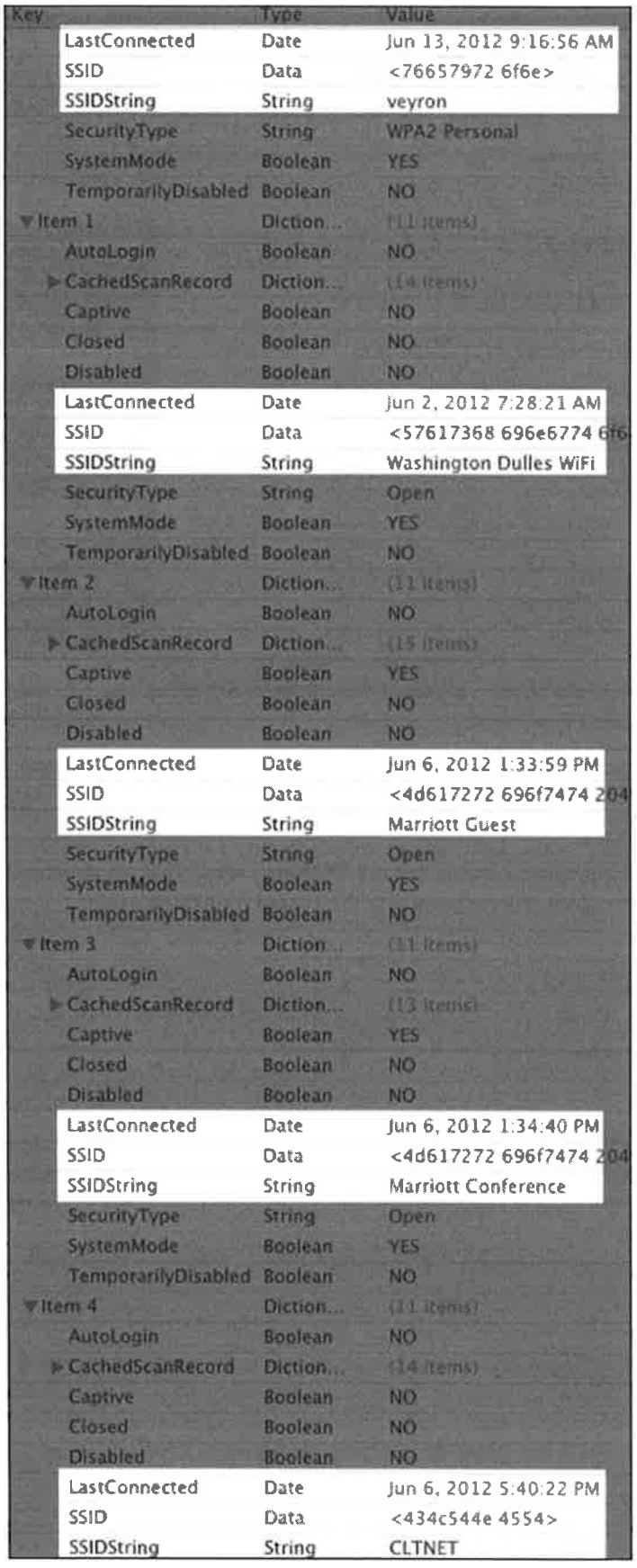
There are many data points in logs showing locational activity, specifically on laptop systems. Laptops are meant to be mobile and to travel around the city, country, or world. The logs can provide a timeline of travel which can help an investigator correlate where a user may have been at a particular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



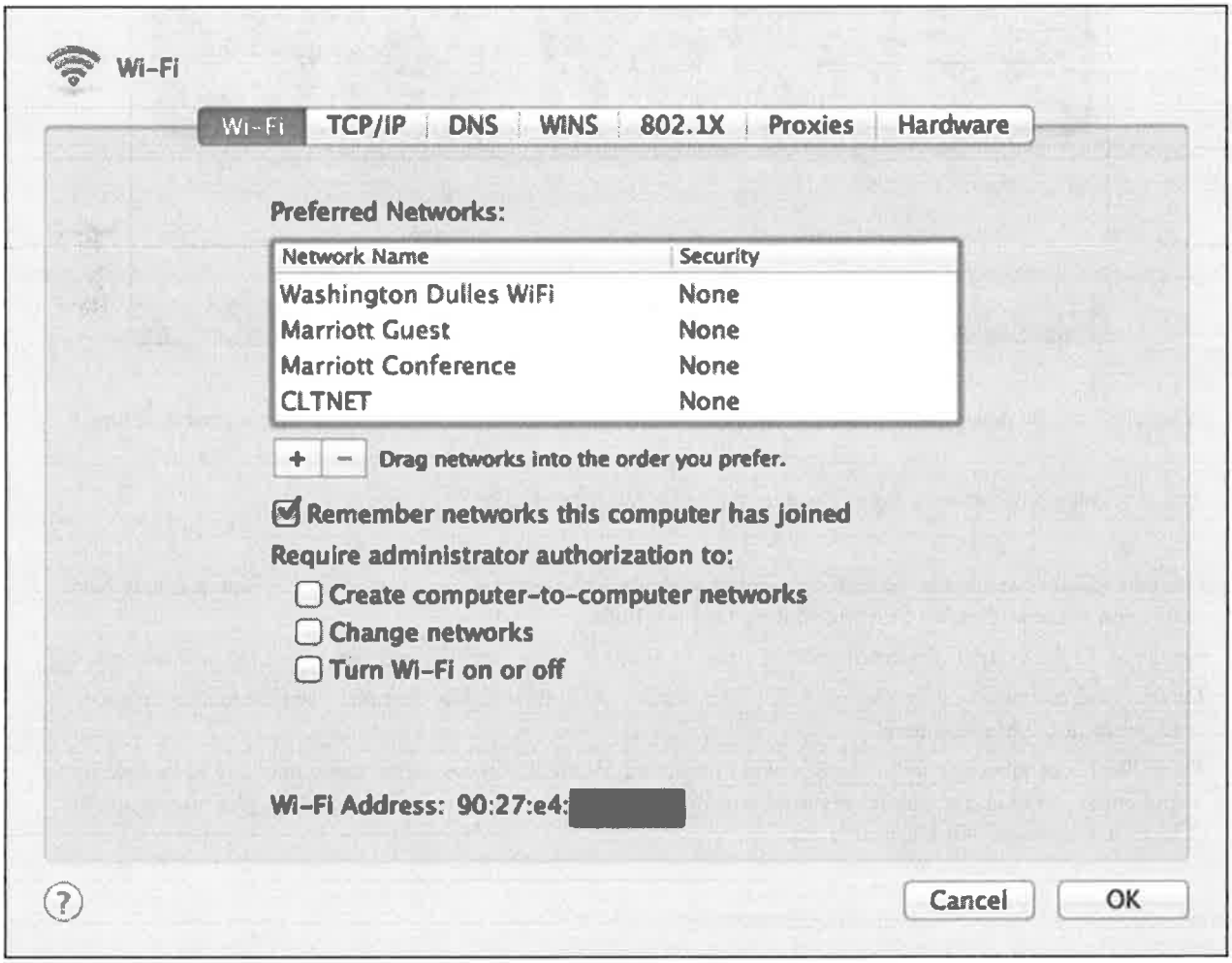
The `com.apple.airport.preferences` property list contains the "remembered" networks. These networks are saved by default on OS X until the user purges them.

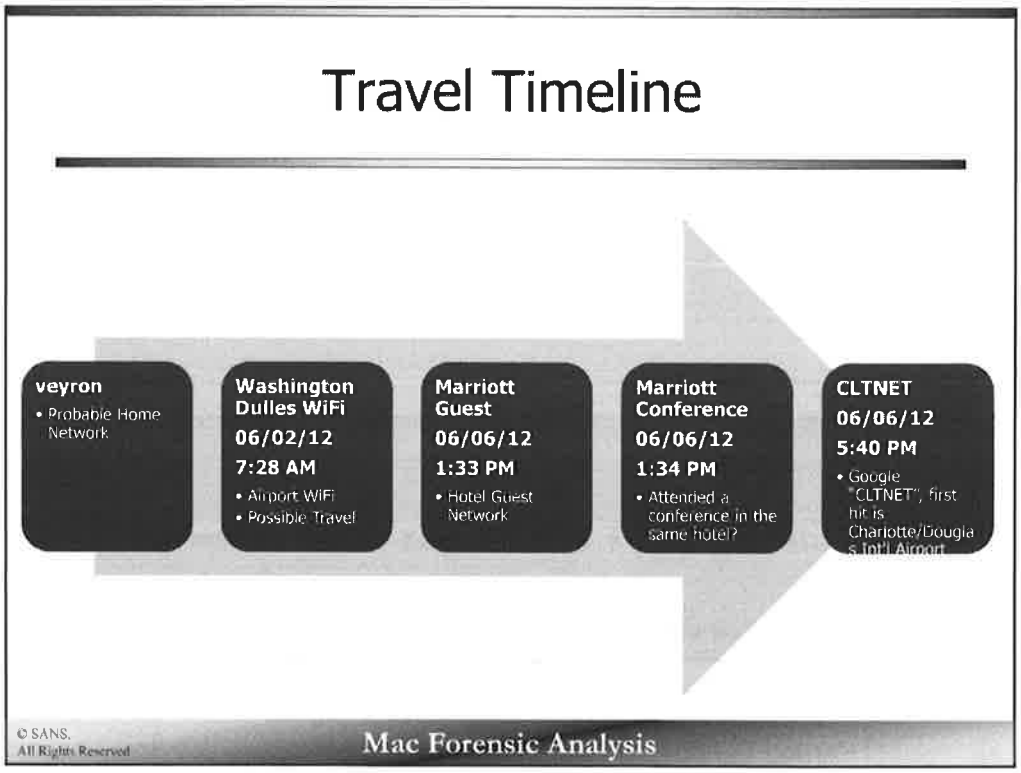
The sidebar shows the raw property list data that the "Preferred Networks" get populated from. The highlighted sections show the last connected time and network SSID string for five networks:

- `veyron`
- `Washington Dulles WiFi`
- `Marriott Guest`
- `Marriott Conference`
- `CLTNET`



Key	Type	Value
LastConnected	Date	Jun 13, 2012 9:16:56 AM
SSID	Data	<76657972 6f6e>
SSIDString	String	veyron
SecurityType	String	WPA2 Personal
SystemMode	Boolean	YES
TemporarilyDisabled	Boolean	NO
▼ Item 1	Diction...	(11 items)
AutoLogin	Boolean	NO
▶ CachedScanRecord	Diction...	(14 items)
Captive	Boolean	NO
Closed	Boolean	NO
Disabled	Boolean	NO
LastConnected	Date	Jun 2, 2012 7:28:21 AM
SSID	Data	<57617368 696e6774 6f6
SSIDString	String	Washington Dulles WiFi
SecurityType	String	Open
SystemMode	Boolean	YES
TemporarilyDisabled	Boolean	NO
▼ Item 2	Diction...	(11 items)
AutoLogin	Boolean	NO
▶ CachedScanRecord	Diction...	(15 items)
Captive	Boolean	YES
Closed	Boolean	NO
Disabled	Boolean	NO
LastConnected	Date	Jun 6, 2012 1:33:59 PM
SSID	Data	<4d617272 696f7474 204
SSIDString	String	Marriott Guest
SecurityType	String	Open
SystemMode	Boolean	YES
TemporarilyDisabled	Boolean	NO
▼ Item 3	Diction...	(11 items)
AutoLogin	Boolean	NO
▶ CachedScanRecord	Diction...	(13 items)
Captive	Boolean	YES
Closed	Boolean	NO
Disabled	Boolean	NO
LastConnected	Date	Jun 6, 2012 1:34:40 PM
SSID	Data	<4d617272 696f7474 204
SSIDString	String	Marriott Conference
SecurityType	String	Open
SystemMode	Boolean	YES
TemporarilyDisabled	Boolean	NO
▼ Item 4	Diction...	(11 items)
AutoLogin	Boolean	NO
▶ CachedScanRecord	Diction...	(14 items)
Captive	Boolean	NO
Closed	Boolean	NO
Disabled	Boolean	NO
LastConnected	Date	Jun 6, 2012 5:40:22 PM
SSID	Data	<434c544e 4554>
SSIDString	String	CLTNET


Wi-Fi
Wi-Fi
TCP/IP
DNS
WINS
802.1X
Proxies
Hardware
Preferred Networks:
Network Name
Security
Washington Dulles WiFi
None
Marriott Guest
None
Marriott Conference
None
CLTNET
None
+
−
Drag networks into the order you prefer.
Remember networks this computer has joined
Require administrator authorization to:
Create computer-to-computer networks
Change networks
Turn Wi-Fi on or off
Wi-Fi Address: 90:27:e4:
Cancel
OK



A timeline can be developed from the networks to show where the device has traveled over a period of time.

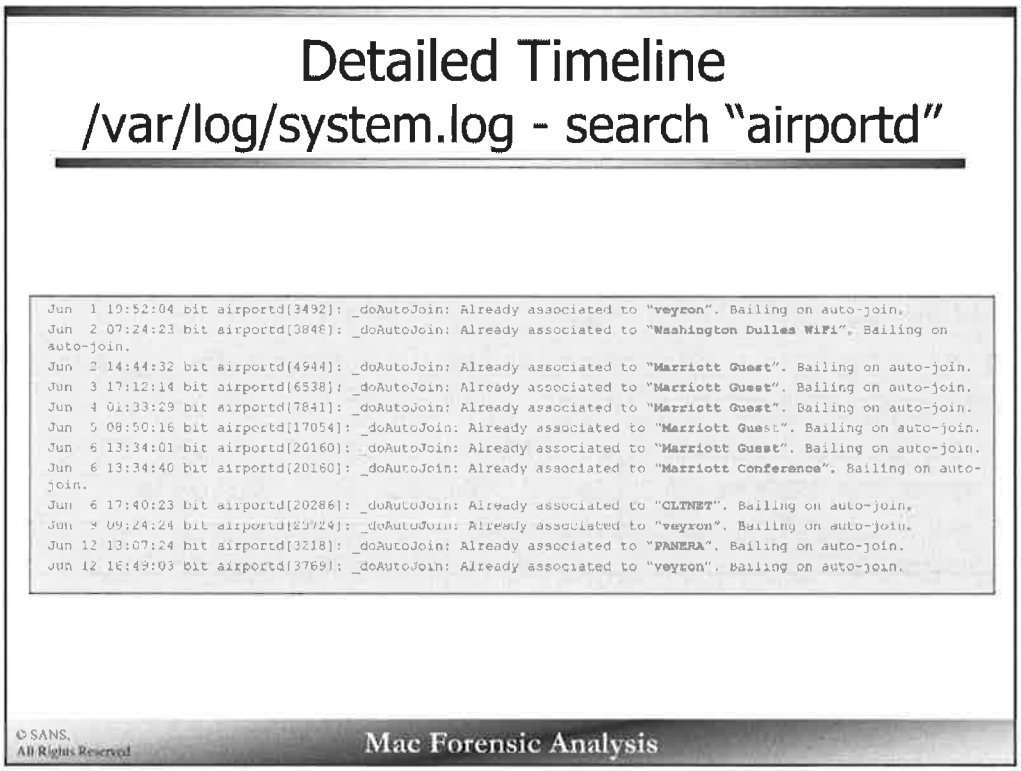
The probable home network for this system has been determined to be "`veyron`".

On 06/02/12 the system can be seen connecting to the "`Washington Dulles WiFi`". The user may have connected to the airport Wi-Fi while waiting for their flight.

On 06/06/12 the system connected to "`Marriott Guest`". This may be the hotel where the user was staying.

On 06/06/12 a connection to "`Marriott Conference`" is shown. The user may have been attending a conference at the Marriott hotel.

On 06/06/12 a connection to "`CLTNET`" was completed. While this access point name may not be as descriptive as the others, a Google search for it shows it as the wireless network used at Charlotte/Douglas International Airport in Charlotte, North Carolina.



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

06/01/12 – Home network '`veyron`'

06/02/12 – Airport Wi-Fi – '`Washington Dulles Airport WiFi`'

06/02/12 – Hotel Wi-Fi– '`Marriott Guest`'
06/03/12 – Hotel Wi-Fi– '`Marriott Guest`'
06/04/12 – Hotel Wi-Fi– '`Marriott Guest`'
06/05/12 – Hotel Wi-Fi– '`Marriott Guest`'
06/06/12 – Hotel Wi-Fi– '`Marriott Guest`'

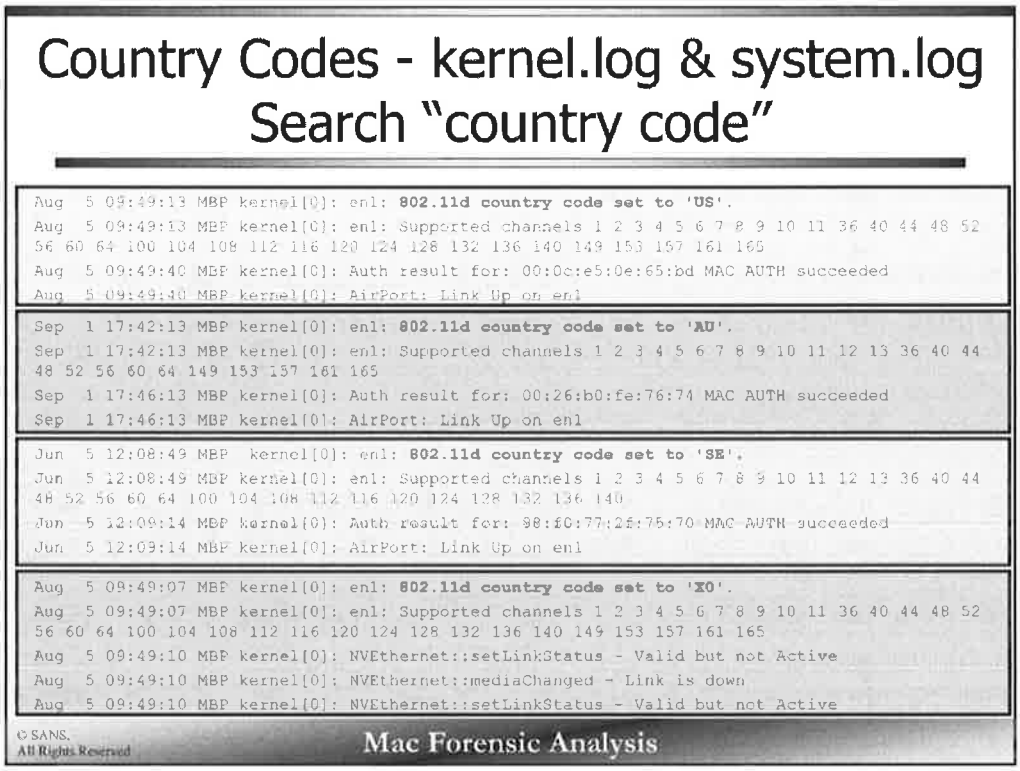
06/06/12 – Conference Wi-Fi– '`Marriott Conference`'

06/06/12 – Airport Wi-Fi– '`CLTNET`'

06/09/12 – Home network – '`veyron`'

06/12/12 – Restaurant – '`PANERA`'
06/12/12 – Home network – '`veyron`'

```
Jun  1 19:52:04 bit airportd[3492]: _doAutoJoin: Already
associated to "veyron". Bailing on auto-join.
Jun  2 07:24:23 bit airportd[3848]: _doAutoJoin: Already
associated to "Washington Dulles WiFi". Bailing on auto-join.
Jun  2 14:44:32 bit airportd[4944]: _doAutoJoin: Already
associated to "Marriott Guest". Bailing on auto-join.
Jun  3 17:12:14 bit airportd[6538]: _doAutoJoin: Already
associated to "Marriott Guest". Bailing on auto-join.
Jun  4 01:33:29 bit airportd[7841]: _doAutoJoin: Already
associated to "Marriott Guest". Bailing on auto-join.
Jun  5 08:50:16 bit airportd[17054]: _doAutoJoin: Already
associated to "Marriott Guest". Bailing on auto-join.
Jun  6 13:34:01 bit airportd[20160]: _doAutoJoin: Already
associated to "Marriott Guest". Bailing on auto-join.
Jun  6 13:34:40 bit airportd[20160]: _doAutoJoin: Already
associated to "Marriott Conference". Bailing on auto-join.
Jun  6 17:40:23 bit airportd[20286]: _doAutoJoin: Already
associated to "CLTNET". Bailing on auto-join.
Jun  9 09:24:24 bit airportd[25724]: _doAutoJoin: Already
associated to "veyron". Bailing on auto-join.
Jun 12 13:07:24 bit airportd[3218]: _doAutoJoin: Already
associated to "PANERA". Bailing on auto-join.
Jun 12 16:49:03 bit airportd[3769]: _doAutoJoin: Already
associated to "veyron". Bailing on auto-join.
```



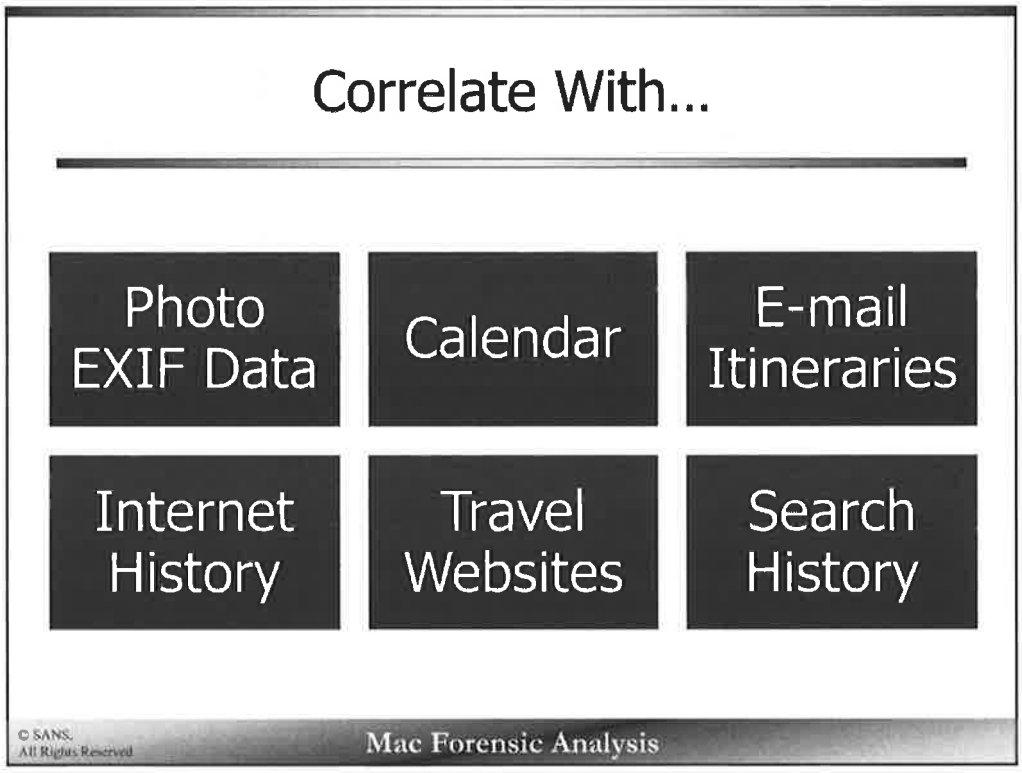
International travel may also be very interesting for your investigation. The country codes for wireless access points are recorded in the `kernel.log` (In 10.7 and before) or the `system.log` (10.8+),

802.11d is an amendment to 802.11 that allows specification on regulatory domains that includes country information to beacons.

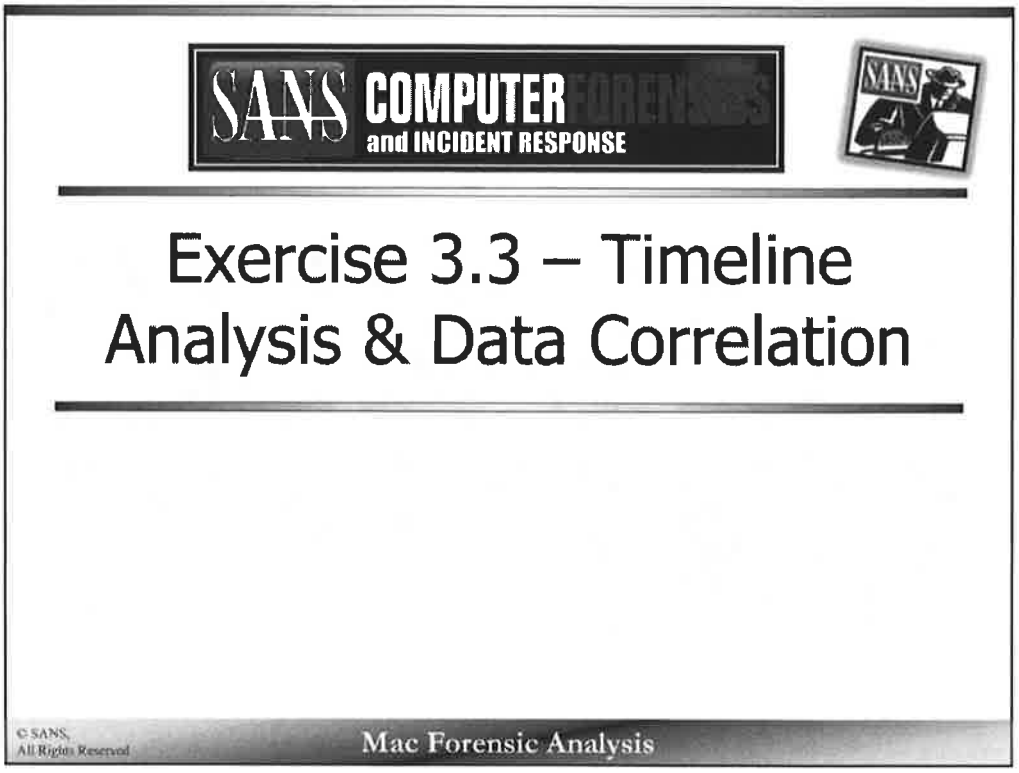
In the example above, the country codes are shown whenever a connection is made to a wireless access point. Each connection is colored in a different color to show related records.

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

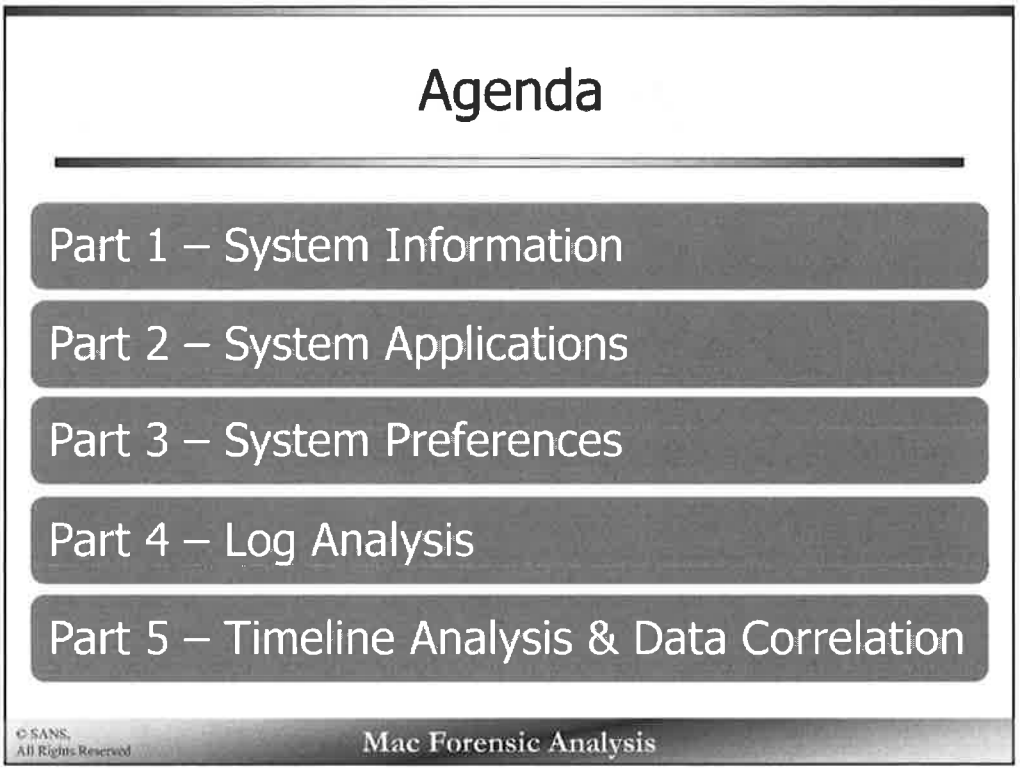
```
Sep  1 17:42:13 MBP kernel[0]: en1: 802.11d country code set to 'AU'.
Sep  1 17:42:13 MBP kernel[0]: en1: Supported channels 1 2 3 4 5 6 7 8 9 10 11
12 13 36 40 44 48 52 56 60 64 149 153 157 161 165
Sep  1 17:46:13 MBP kernel[0]: Auth result for: 00:26:b0:fe:76:74 MAC AUTH
succeeded
Sep  1 17:46:13 MBP kernel[0]: AirPort: Link Up on en1
…
Aug  5 09:49:07 MBP kernel[0]: en1: 802.11d country code set to 'X0'.
Aug  5 09:49:07 MBP kernel[0]: en1: Supported channels 1 2 3 4 5 6 7 8 9 10 11
36 40 44 48 52 56 60 64 100 104 108 112 116 120 124 128 132 136 140 149 153
157 161 165
Aug  5 09:49:10 MBP kernel[0]: NVEthernet::setLinkStatus - Valid but not
Active
Aug  5 09:49:10 MBP kernel[0]: NVEthernet::mediaChanged - Link is down
Aug  5 09:49:10 MBP kernel[0]: NVEthernet::setLinkStatus - Valid but not
Active
Aug  5 09:49:13 MBP kernel[0]: en1: 802.11d country code set to 'US'.
Aug  5 09:49:13 MBP kernel[0]: en1: Supported channels 1 2 3 4 5 6 7 8 9 10 11
36 40 44 48 52 56 60 64 100 104 108 112 116 120 124 128 132 136 140 149 153
157 161 165
Aug  5 09:49:40 MBP kernel[0]: Auth result for: 00:0c:e5:0e:65:bd MAC AUTH
succeeded
Aug  5 09:49:40 MBP kernel[0]: AirPort: Link Up on en1
…
Jun  5 12:08:49 MBP  kernel[0]: en1: 802.11d country code set to 'SE'.
Jun  5 12:08:49 MBP kernel[0]: en1: Supported channels 1 2 3 4 5 6 7 8 9 10 11
12 13 36 40 44 48 52 56 60 64 100 104 108 112 116 120 124 128 132 136 140
Jun  5 12:09:14 MBP kernel[0]: Auth result for: 88:f0:77:2f:75:70 MAC AUTH
succeeded
Jun  5 12:09:14 MBP kernel[0]: AirPort: Link Up on en1
```



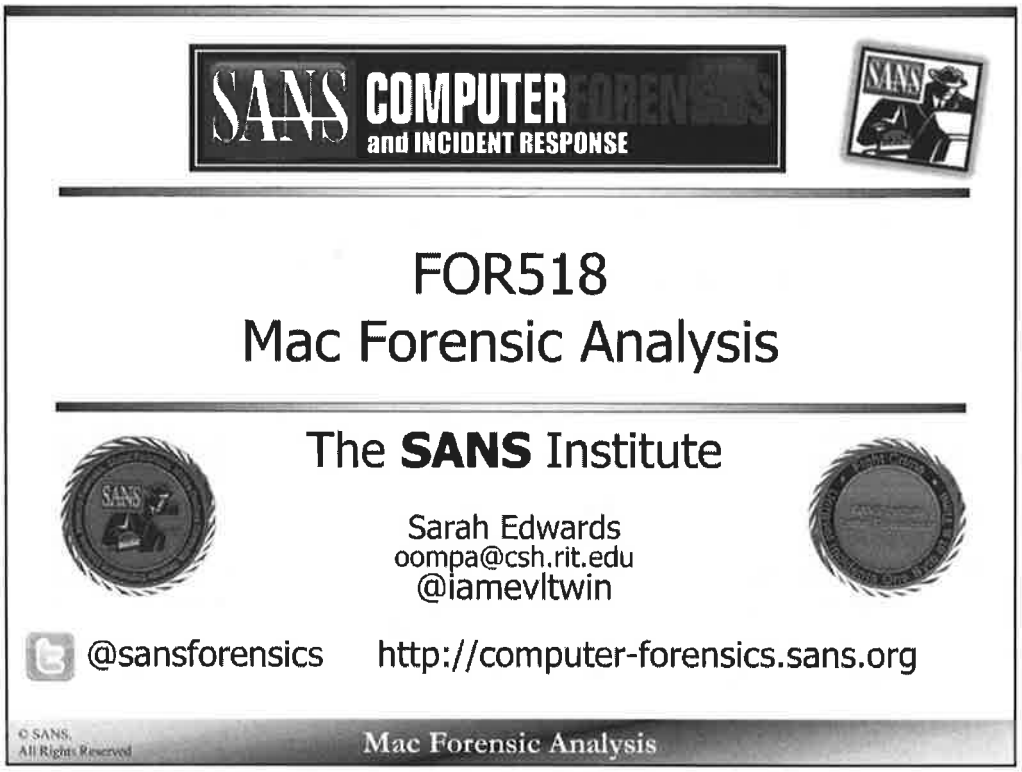
The travel data found in the logs can be correlated with many other forensic artifacts.



This page intentionally left blank.



This page intentionally left blank.



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