SANS

www.sans.org

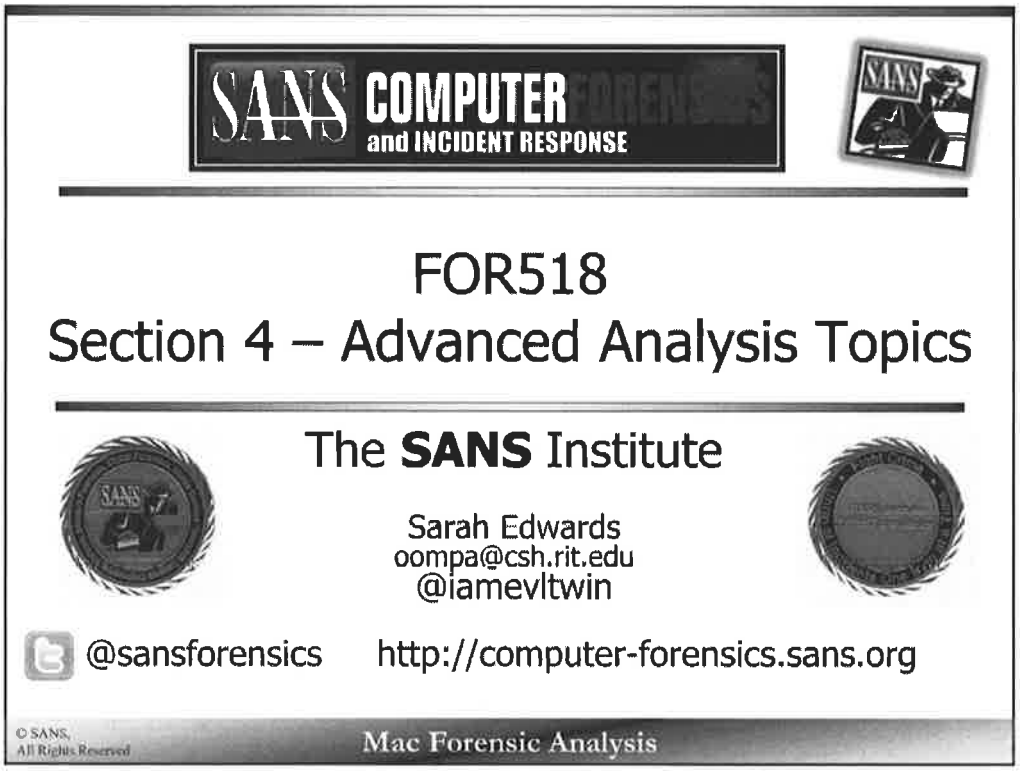
FORENSICS 518

MAC FORENSIC ANALYSIS

518.4

Advanced Analysis Topics

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



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

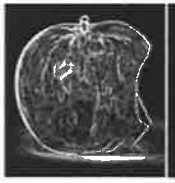
FOR508
Advanced Incident Response
GCFA

FOR572
Advanced Network Forensics and Analysis
GNFA

LEARN REM

FOR610
REM: Malware Analysis
GREM

SPECIALIZATION

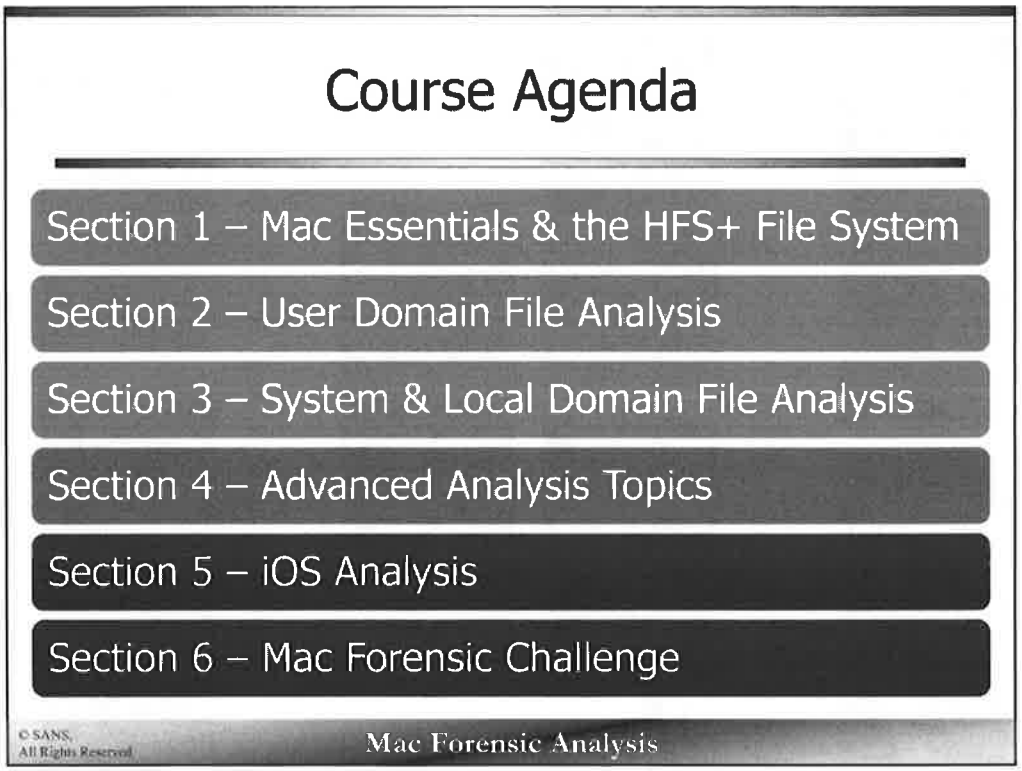


FOR518
Mac Forensics

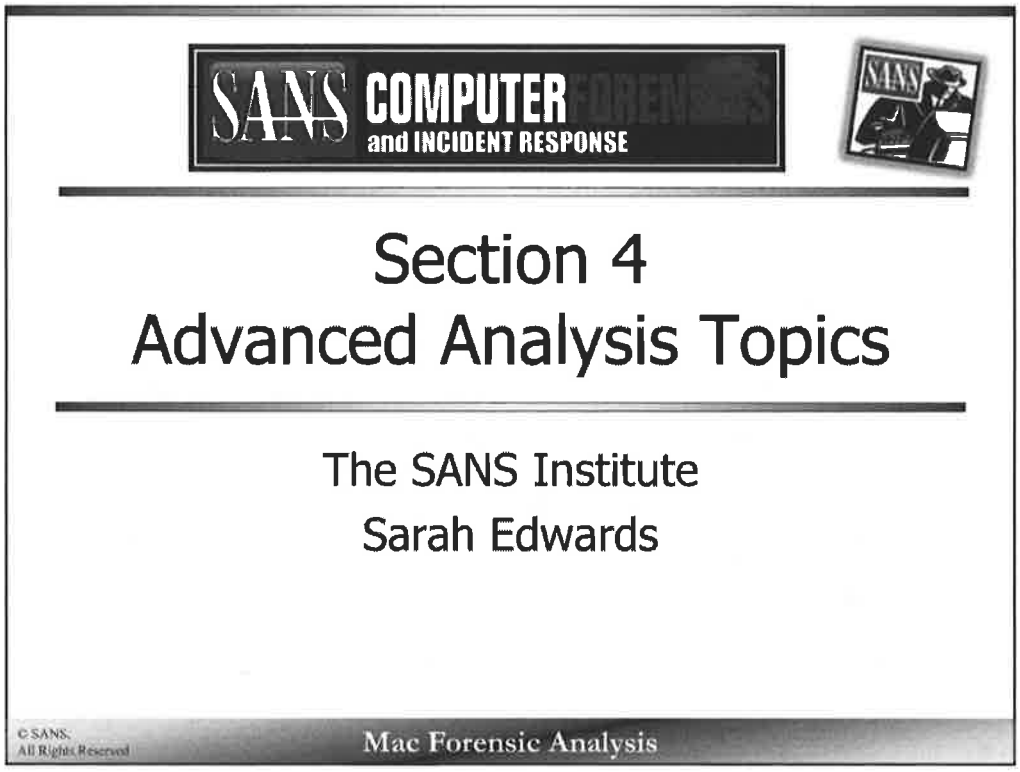
FOR526
Memory Forensics In-Depth

MGT535
Incident Response Team Management

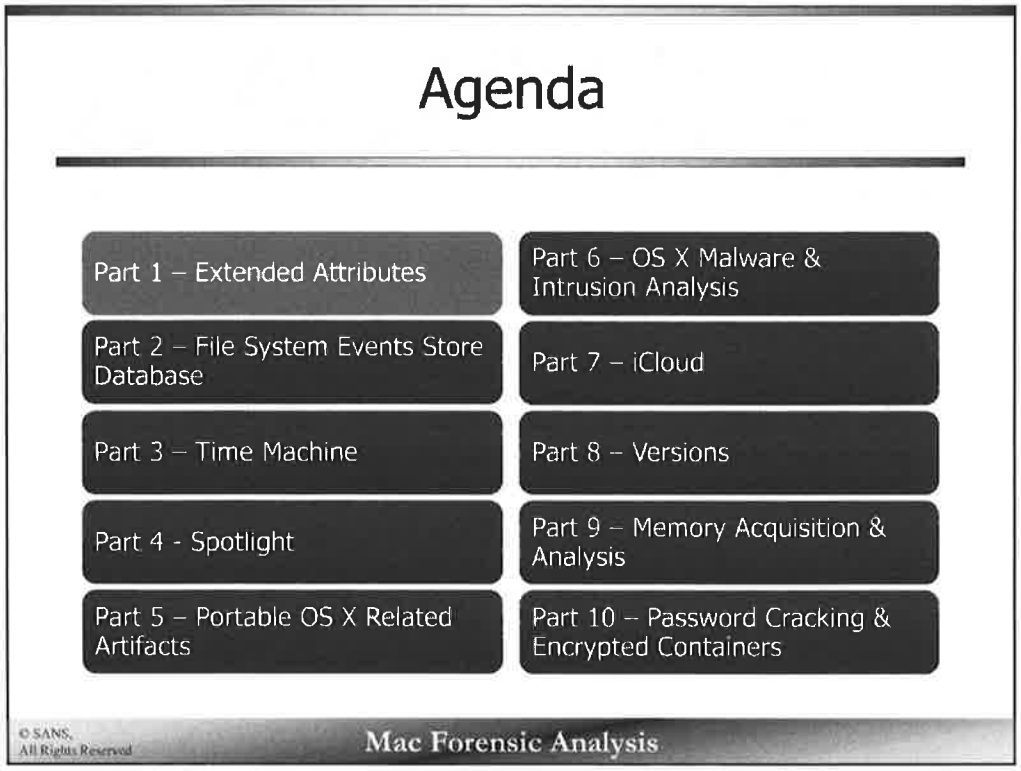
FOR585
Advanced Smartphone Forensics



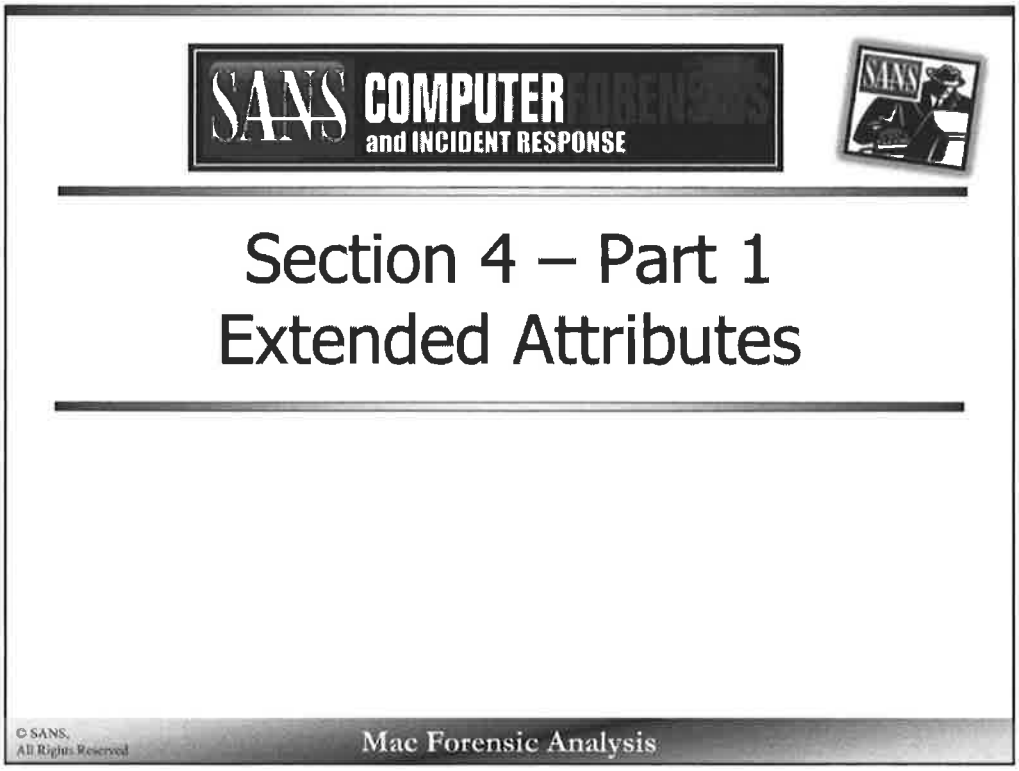
This page intentionally left blank.



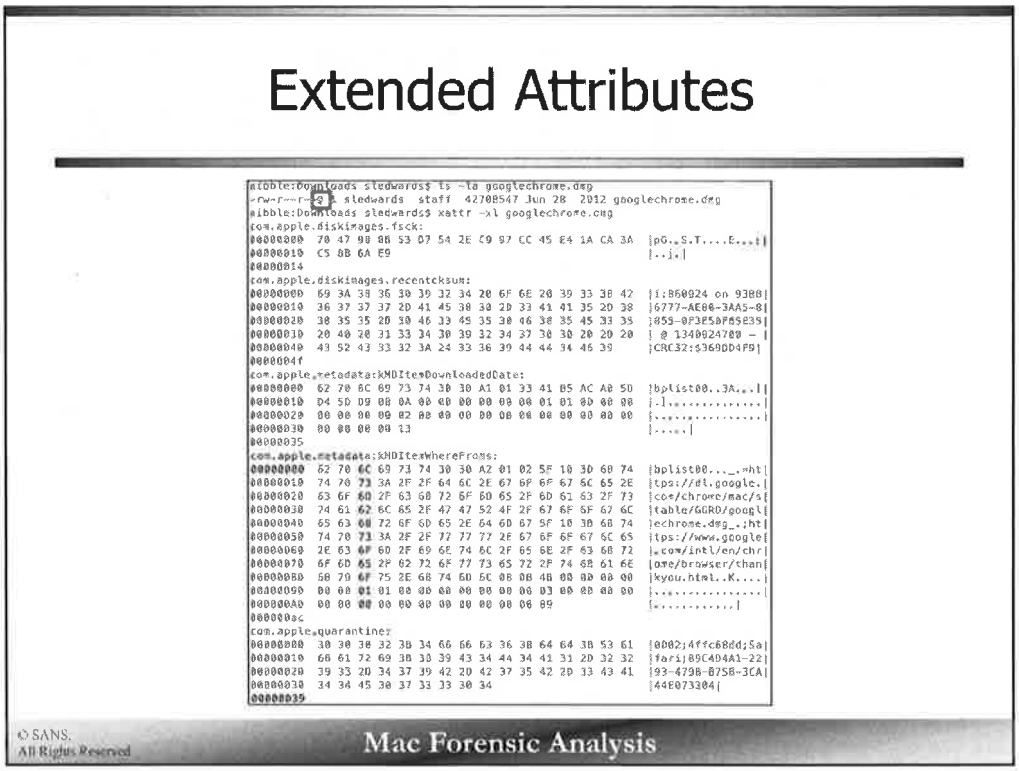
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.



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

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

The screenshot above shows the output of the `ls -la` command on the file `googlechrome.dmg`. This output contains a "`@`" in the file mode section of a long file listing which tells the user there are extended attributes for this file.

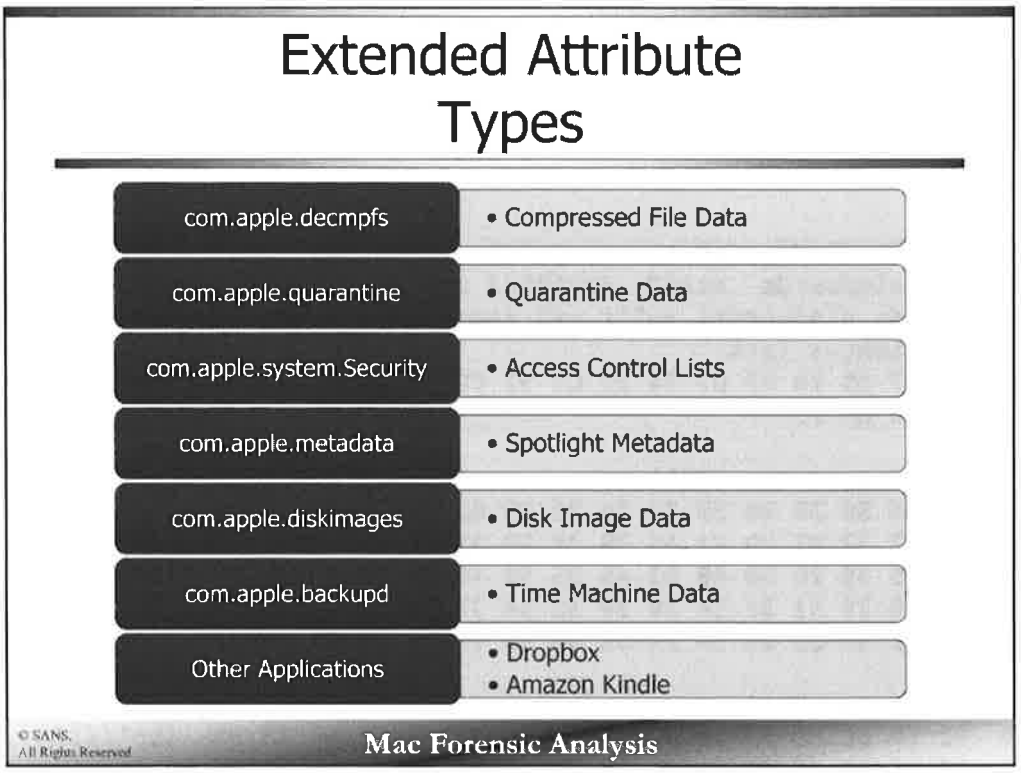
The next command, `xattr -xl googlechrome.dmg`, allows us to view the extended attributes for this file. Each extended attribute is named using reverse DNS format (i.e. com.apple.quarantine).

References:

Mac OS X and iOS Internals: To the Apple's Core

Chapter 16 – HFS+ File System Concepts

```
nibble:Downloads sledwards$ ls -la googlechrome.dmg
-rw-r--r--@ 1 sledwards  staff  42708547 Jun 28  2012 googlechrome.dmg
nibble:Downloads sledwards$ xattr -xl googlechrome.dmg
com.apple.diskimages.fsck:
00000000  70 47 98 8B 53 D7 54 2E C9 97 CC 45 E4 1A CA 3A  |pG..S.T....E...:|
00000010  C5 8B 6A E9                                      |..j.|
00000014
com.apple.diskimages.recentcksum:
00000000  69 3A 38 36 30 39 32 34 20 6F 6E 20 39 33 38 42  |i:860924 on 9388|
00000010  36 37 37 37 2D 41 45 38 30 2D 33 41 41 35 2D 38  |6777-AE80-3AA5-8|
00000020  38 35 35 2D 30 46 33 45 35 30 46 38 35 45 33 35  |855-0F3E50F85E35|
00000030  20 40 20 31 33 34 30 39 32 34 37 30 30 20 2D 20  | @ 1340924700 - |
00000040  43 52 43 33 32 3A 24 33 36 39 44 44 34 46 39     |CRC32:$369DD4F9|
0000004f
com.apple.metadata:kMDItemDownloadedDate:
00000000  62 70 6C 69 73 74 30 30 A1 01 33 41 B5 AC A0 5D  |bplist00..3A...]|
00000010  D4 5D D9 08 0A 00 00 00 00 00 00 01 01 00 00 00  |.]..............|
00000020  00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 13                                   |.....|
00000035
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 3D 68 74  |bplist00..._.=ht|
00000010  74 70 73 3A 2F 2F 64 6C 2E 67 6F 6F 67 6C 65 2E  |tps://dl.google.|
00000020  63 6F 6D 2F 63 68 72 6F 6D 65 2F 6D 61 63 2F 73  |com/chrome/mac/s|
00000030  74 61 62 6C 65 2F 47 47 52 4F 2F 67 6F 6F 67 6C  |table/GGRO/googl|
00000040  65 63 68 72 6F 6D 65 2E 64 6D 67 5F 10 3B 68 74  |echrome.dmg_.;ht|
00000050  74 70 73 3A 2F 2F 77 77 77 2E 67 6F 6F 67 6C 65  |tps://www.google|
00000060  2E 63 6F 6D 2F 69 6E 74 6C 2F 65 6E 2F 63 68 72  |.com/intl/en/chr|
00000070  6F 6D 65 2F 62 72 6F 77 73 65 72 2F 74 68 61 6E  |ome/browser/than|
00000080  6B 79 6F 75 2E 68 74 6D 6C 08 0B 4B 00 00 00 00  |kyou.html..K....|
00000090  00 00 01 01 00 00 00 00 00 00 00 03 00 00 00 00  |................|
000000A0  00 00 00 00 00 00 00 00 00 00 00 89              |............|
000000ac
com.apple.quarantine:
00000000  30 30 30 32 3B 34 66 66 63 36 38 64 64 3B 53 61  |0002;4ffc68dd;Sa|
00000010  66 61 72 69 3B 38 38 39 43 34 44 34 41 31 2D 32 32  |fari;89C4D4A1-22|
00000020  39 33 2D 34 37 39 42 2D 42 37 35 42 2D 33 43 41  |93-479B-B75B-3CA|
00000030  34 34 45 30 37 33 33 30 34                       |44E073304|
00000039
```

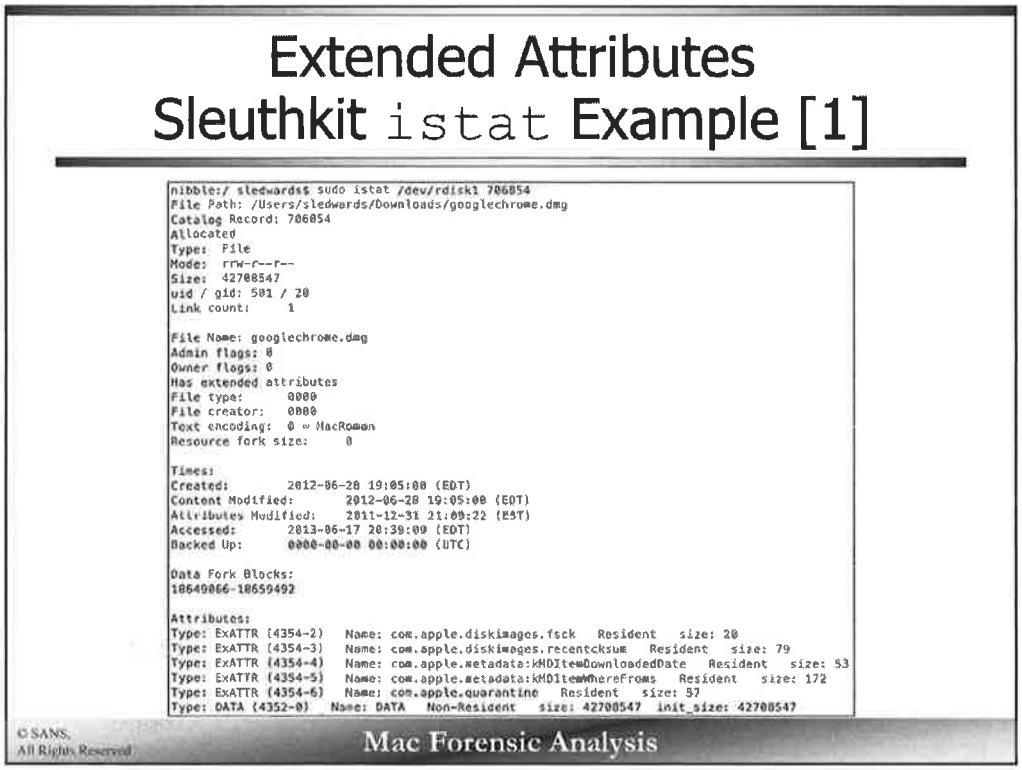


Extended attributes come in many types, some of which are listed above. The two most popular are `com.apple.decmpfs` and `com.apple.quarantine`. Each attribute contains data specific to its purpose and does not follow a static format. An investigator's intuition and reverse engineering skills may be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

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

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

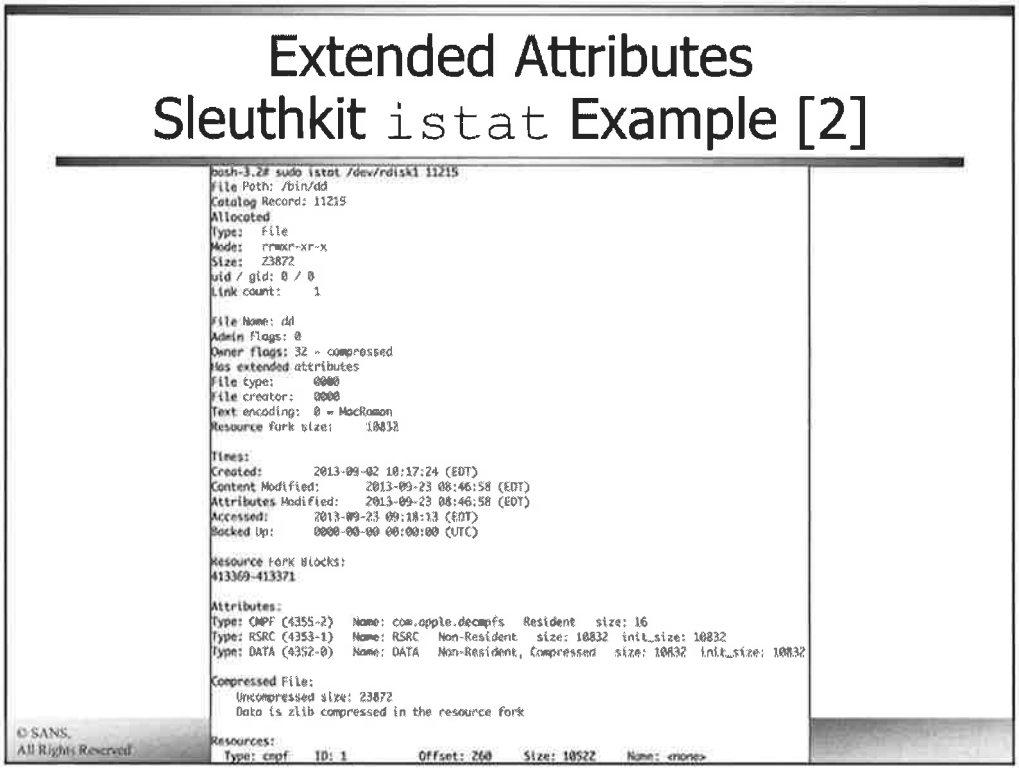
Note: The `xattr` command filters out `com.apple.decmpfs` and `com.apple.system.Security` attributes. An investigator may need additional tools to review these two attributes.



The Sleuthkit's (TSK) `istat` command prints the name and sizes of extended attributes in its "`Attributes`" section shown in the screenshot above for the file with CNID (inode) 706854.

Each attribute contains a TSK assigned attribute. For extended attributes this number is `4354-#`, where # is filled in for each individual attribute.

```
nibble:/ sledwards$ sudo istat /dev/rdisk1 706854
File Path: /Users/sledwards/Downloads/googlechrome.dmg
Catalog Record: 706854
Allocated
Type:  File
Mode:  rrw-r--r--
Size:  42708547
uid / gid: 501 / 20
Link count:     1

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

Times:
Created:        2012-06-28 19:05:00 (EDT)
Content Modified:       2012-06-28 19:05:00 (EDT)
Attributes Modified:    2011-12-31 21:09:22 (EST)
Accessed:       2013-06-17 20:39:09 (EDT)
Backed Up:      0000-00-00 00:00:00 (UTC)

Data Fork Blocks:
18649066-18659492

Attributes:
Type: ExATTR (4354-2)   Name: com.apple.diskimages.fsck   Resident   size: 20
Type: ExATTR (4354-3)   Name: com.apple.diskimages.recentcksum   Resident   size: 79
Type: ExATTR (4354-4)   Name: com.apple.metadata:kMDItemDownloadedDate   Resident   size: 53
Type: ExATTR (4354-5)   Name: com.apple.metadata:kMDItemWhereFroms   Resident   size: 172
Type: ExATTR (4354-6)   Name: com.apple.quarantine   Resident   size: 57
Type: DATA (4352-0)   Name: DATA   Non-Resident   size: 42708547  init_size: 42708547
```

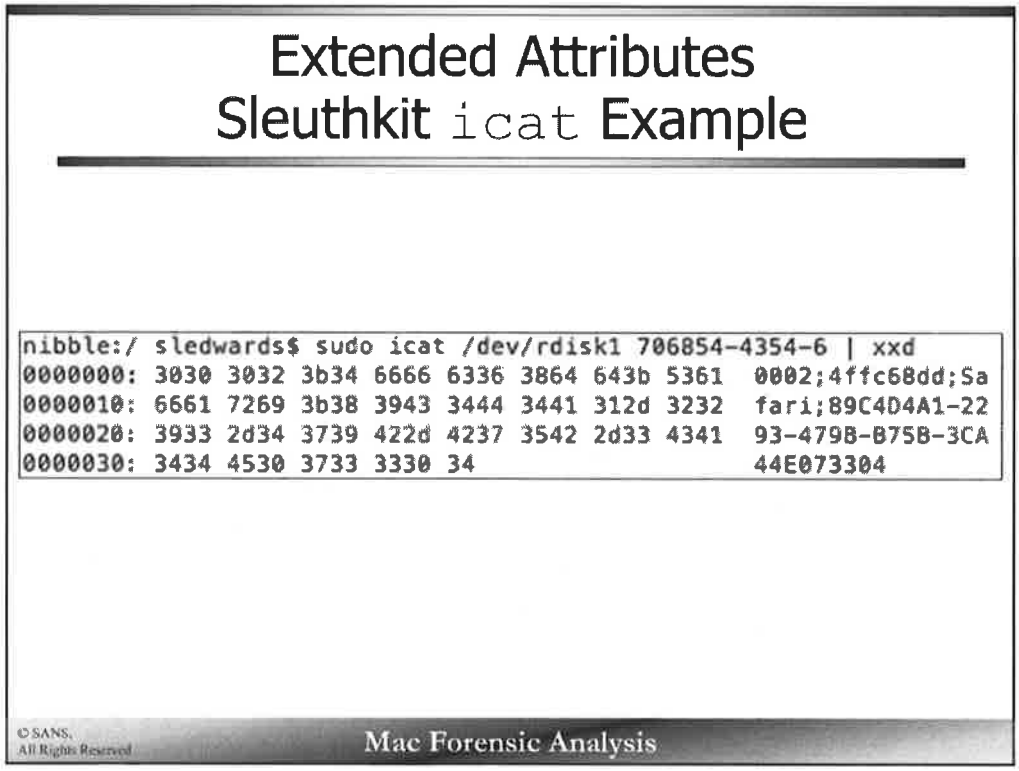


This screenshot shows an example of the `/bin/dd` (CNID 11215) utility where the HFS+ file compression is used.

Note the attributes types:

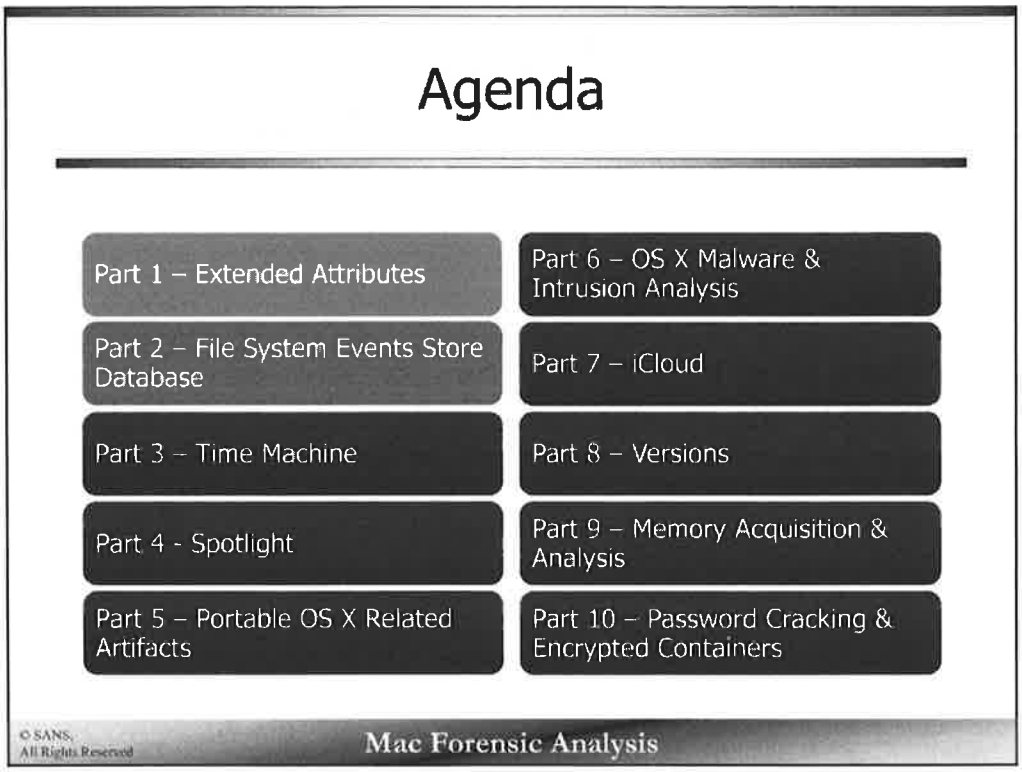
- CMPF – Compressed File Data (com.apple.decmpfs stored in the $Attributes file)
- RSRC – Resource Fork
- DATA – Data Fork

```
bash-3.2# sudo istat /dev/rdisk1 11215
File Path: /bin/dd
Catalog Record: 11215
Allocated
Type:    File
Mode:    rrwxr-xr-x
Size:    23872
uid / gid: 0 / 0
Link count:      1

File Name: dd
Admin flags: 0
Owner flags: 32 - compressed
Has extended attributes
File type:       0000
File creator:    0000
Text encoding:   0 = MacRoman
Resource fork size:      10832

Times:
Created:         2013-09-02 10:17:24 (EDT)
Content Modified:        2013-09-23 08:46:58 (EDT)
Attributes Modified:     2013-09-23 08:46:58 (EDT)
Accessed:        2013-09-23 09:18:13 (EDT)
Backed Up:       0000-00-00 00:00:00 (UTC)

Resource Fork Blocks:
413369-413371

Attributes:
Type: CMPF (4355-2)   Name: com.apple.decmpfs   Resident   size: 16
Type: RSRC (4353-1)   Name: RSRC   Non-Resident   size: 10832  init_size: 10832
Type: DATA (4352-0)   Name: DATA   Non-Resident, Compressed   size: 10832  init_size: 10832

Compressed File:
    Uncompressed size: 23872
    Data is zlib compressed in the resource fork

Resources:
  Type: cmpf    ID: 1          Offset: 260     Size: 10522     Name: <none>
```

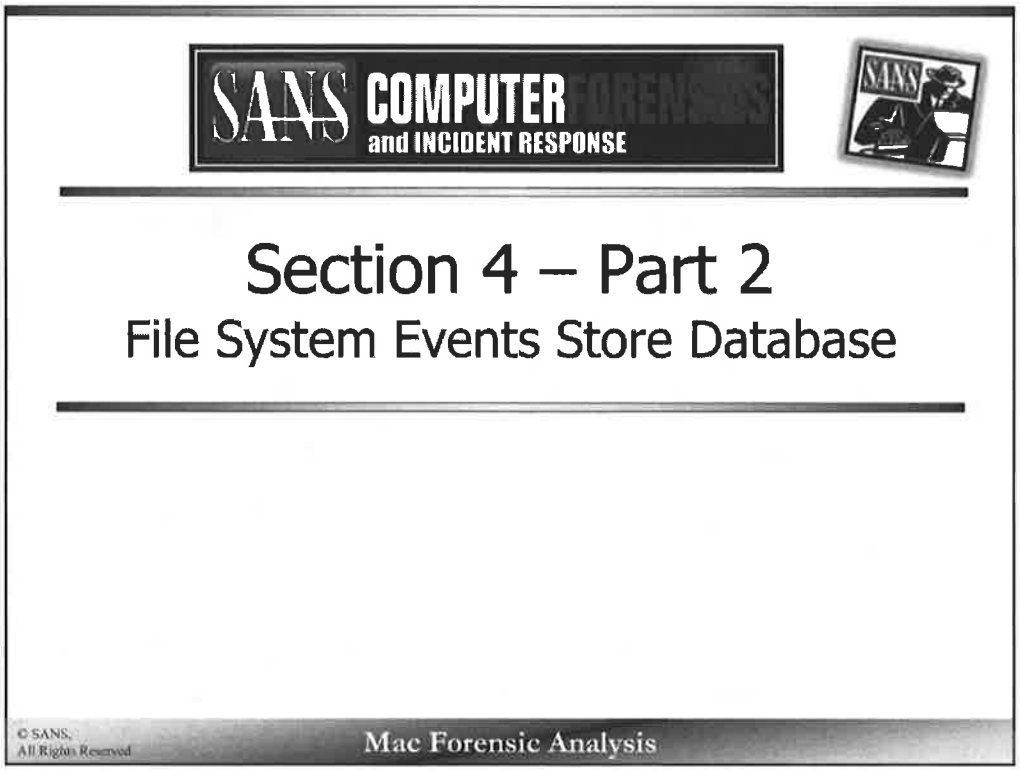


The Sleuthkit `icat` command can be used to print the data contained in a specific attribute. The TSK attribute number is appended to the CNID (inode) of the file to print the specific attribute.

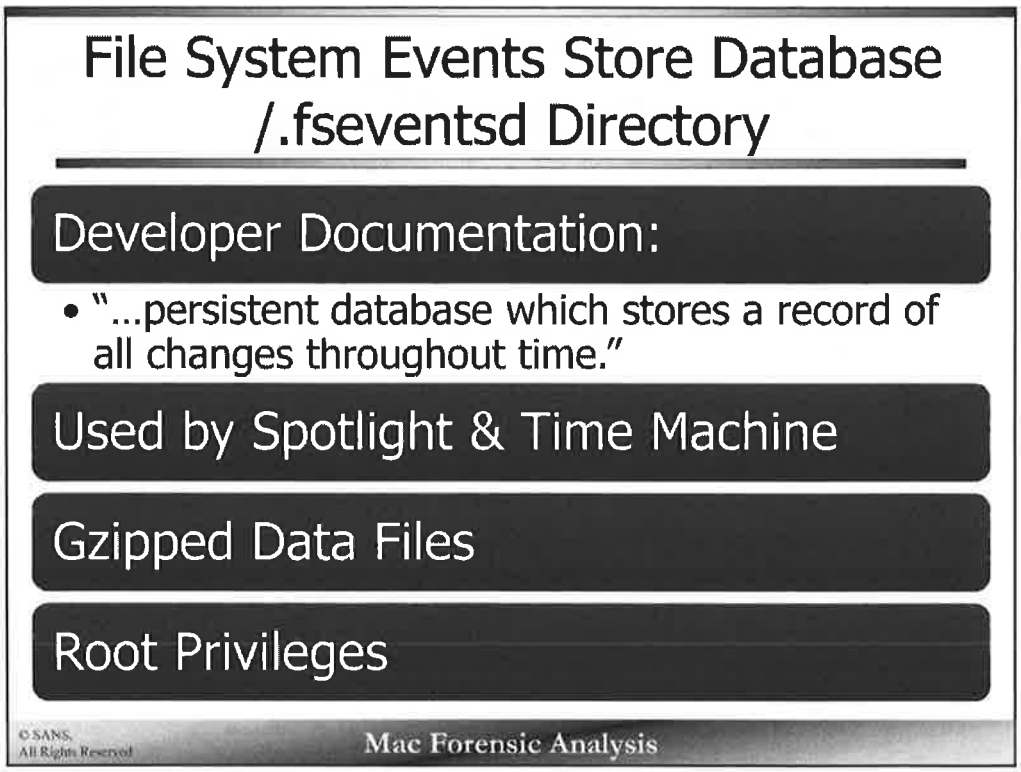
The screenshot shows the `icat` command printing the extended attributes identified by the `4354-6` for the CNID `706854`. This is the `com.apple.quarantine` attribute.



This page intentionally left blank.



This page intentionally left blank.



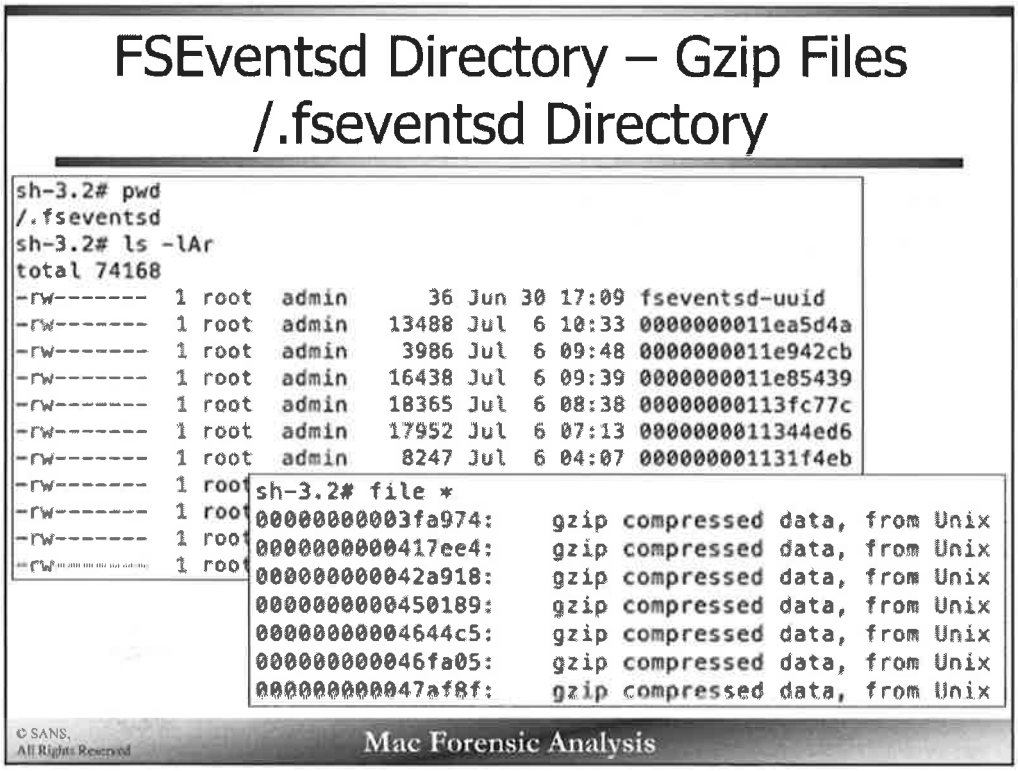
Each volume connected to a Mac system will have a `/.fseventsd` directory created. This directory is the File System Events Store Database which is responsible for storing file system changes on the volume.

Spotlight and Time Machine use this database to determine what files are new or have changed metadata properties. This directory contains gzipped files that require root privileges to unzip and view them.

References:

Apple Developer Documentation - File System Events Programming Guide

https://developer.apple.com/library/mac/#documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005289-CH1-SW1



The `/.fseventsd` directory contains gzipped files each named incrementally with no file extension.

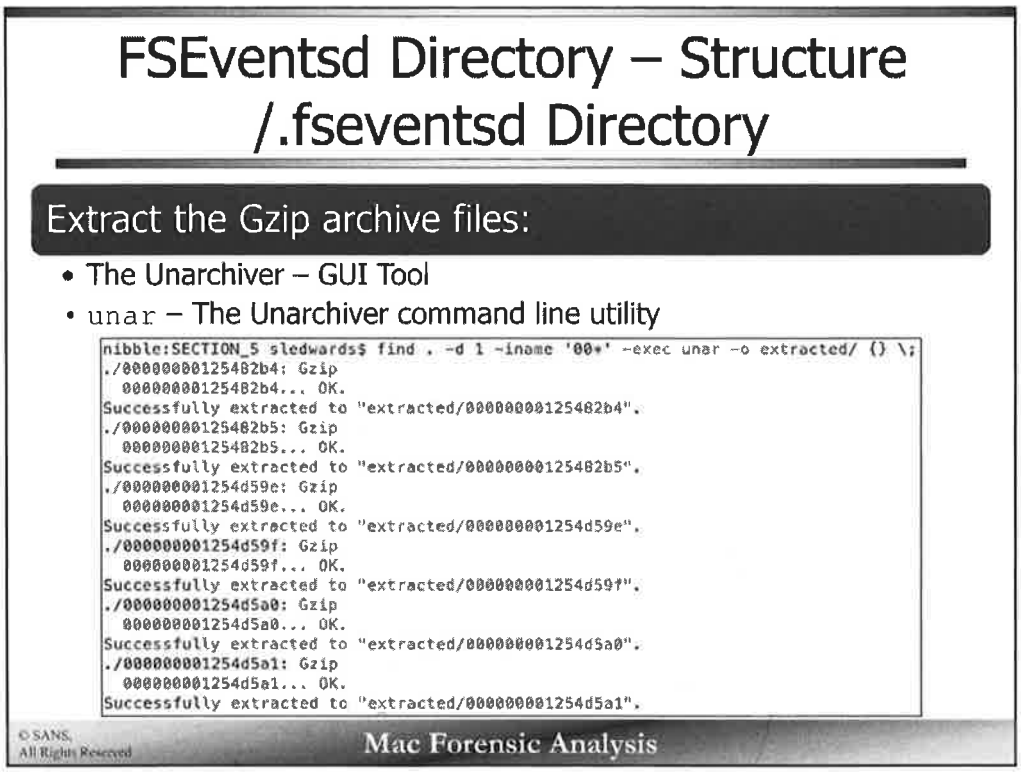
The `fseventsd-uuid` file contains the GUID of the FSEvents database. On a HDD volume this should stay persistent, barring system malfunction, while on a external USB drive it is likely to change each time the drive is inserted into a system. FSEvent UUIDs will show changes in the system logs if you do a search for "`fsevents`".

The /.fseventsd directory will be created on non-HFS+ volumes; however, it does not appear to create the database files.

References:

Apple Developer Documentation - File System Events Programming Guide

https://developer.apple.com/library/mac/#documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005289-CH1-SW1



To explore the fsevents database files, we will need to unzip them.

My preferred tool to extract just about any archive is a free tool called The Unarchiver. This tool also has a command line equivalent called `unar`. These tools are available at `http://unarchiver.c3.cx/commandline`. While there are native tools for OS X that can decompress these archives, the author finds the Unarchiver more extensible. For example, we can use the native tool `gunzip` to decompress gzip files; however, there is no option to output the decompressed files to another directory.

The following command line can be used to extract the contents of all the FSEvents database files (those files starting with "`00`"). This command line uses the find command to traverse a depth of 1 (`-d 1`) searching for filenames starting with "`00`" (`-iname '00*'`) and executes the `unar` utility which outputs the extracted files to the `/extracted` directory (`-exec unar -o extracted/ {} \;`).

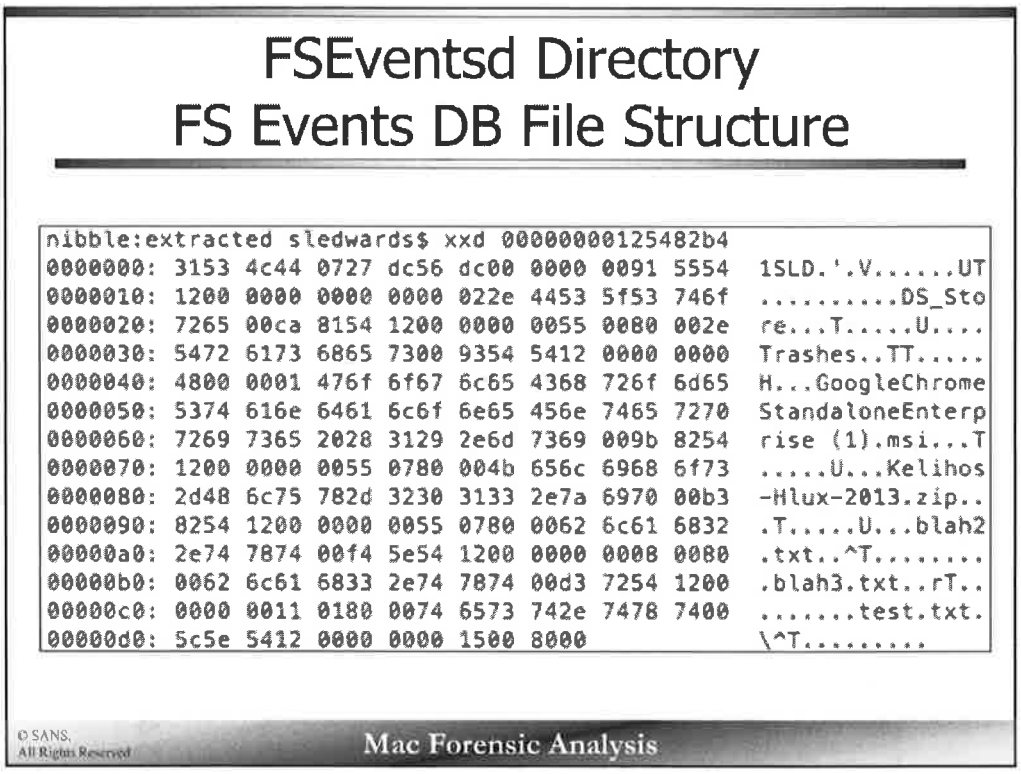
```
find . -d 1 -iname '00*' -exec unar -o extracted/ {} \;
```

References:

Apple Developer Documentation - File System Events Programming Guide

https://developer.apple.com/library/mac/#documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005289-CH1-SW1

```
nibble:SECTION_5 sledwards$ find . -d 1 -iname '00*' -exec unar -o extracted/ {} \;
./00000000125482b4: Gzip
  00000000125482b4... OK.
Successfully extracted to "extracted/00000000125482b4".
./00000000125482b5: Gzip
  00000000125482b5... OK.
Successfully extracted to "extracted/00000000125482b5".
./000000001254d59e: Gzip
  000000001254d59e... OK.
Successfully extracted to "extracted/000000001254d59e".
./000000001254d59f: Gzip
  000000001254d59f... OK.
Successfully extracted to "extracted/000000001254d59f".
./000000001254d5a0: Gzip
  000000001254d5a0... OK.
Successfully extracted to "extracted/000000001254d5a0".
./000000001254d5a1: Gzip
  000000001254d5a1... OK.
Successfully extracted to "extracted/000000001254d5a1".
```



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

References:

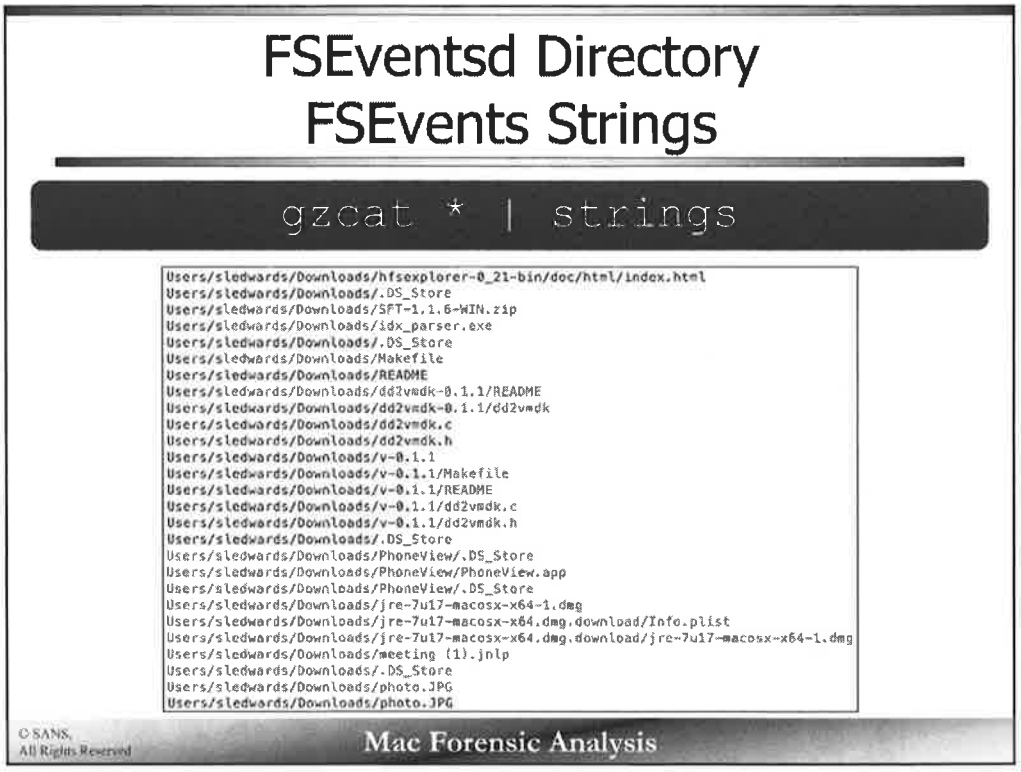
Apple Developer Documentation - File System Events Programming Guide

https://developer.apple.com/library/mac/#documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005289-CH1-SW1

Apple Developer Documentation - FSEvents Reference

http://developer.apple.com/library/mac/#documentation/Darwin/Reference/FSEvents_Ref/Reference/reference.html

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



The screenshot above shows an example of the contents of using strings on the extracted FSEvents Database files. To give this example context, this a only a few lines of the some three million lines extracted from the FSEvents database on my current system.

You may also want to use the `gzcat * | strings` command to get the same information without extracting all the files.

References:

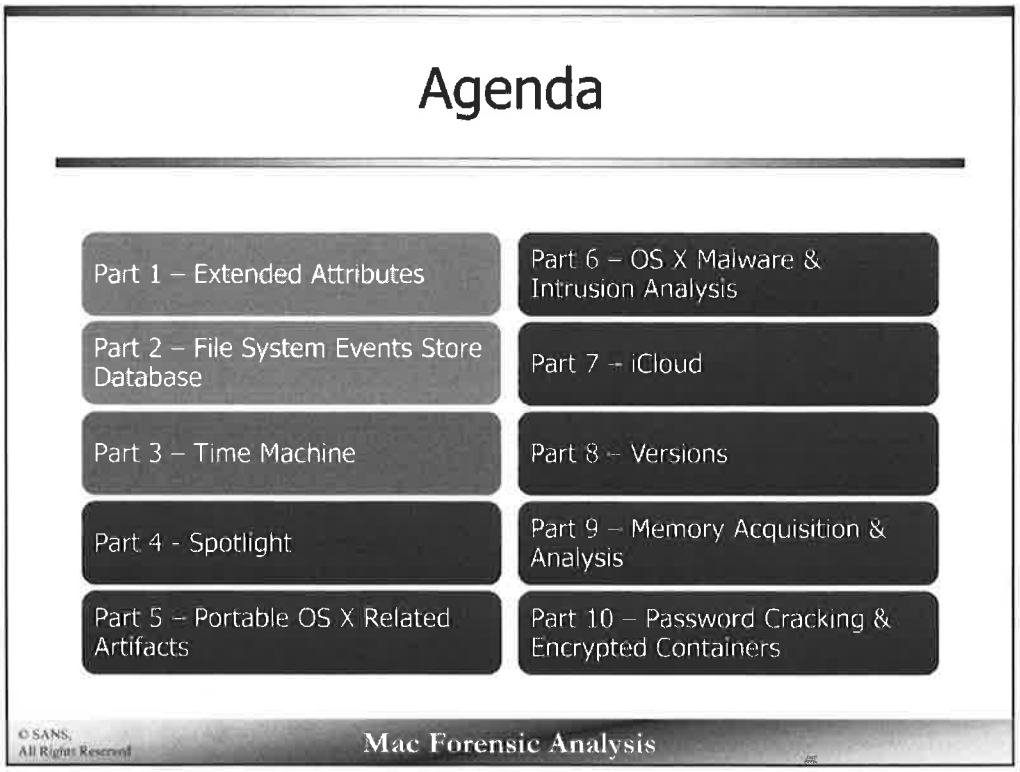
Apple Developer Documentation - File System Events Programming Guide

https://developer.apple.com/library/mac/#documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005289-CH1-SW1

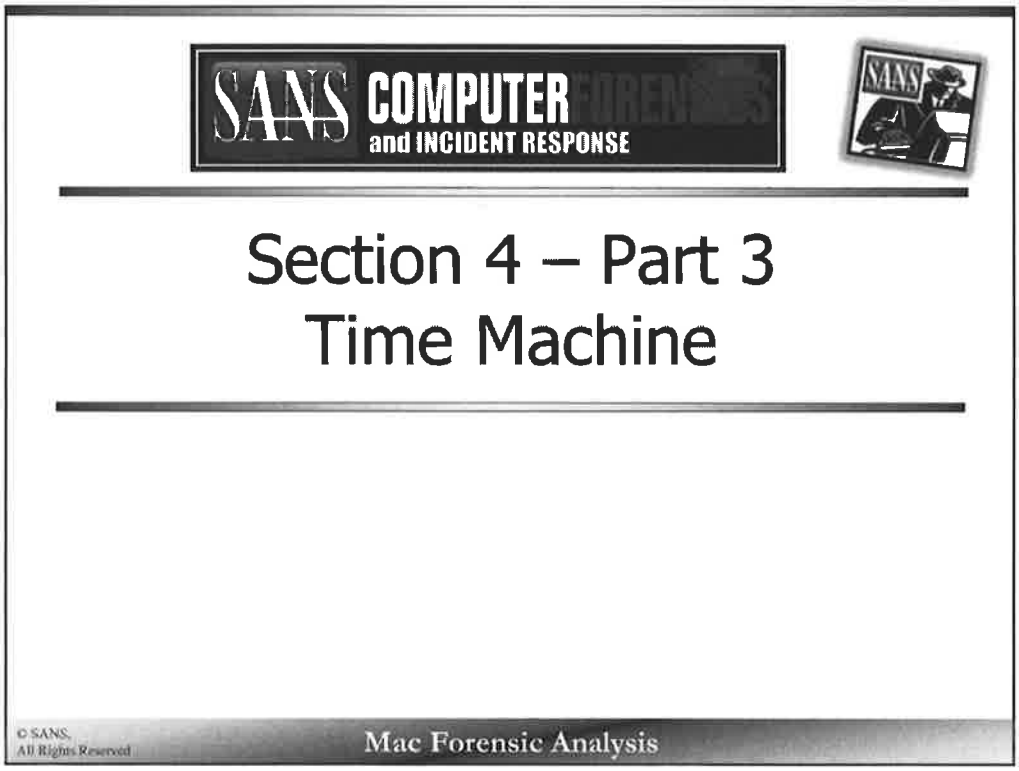
Apple Developer Documentation - FSEvents Reference

http://developer.apple.com/library/mac/#documentation/Darwin/Reference/FSEvents_Ref/Reference/reference.html

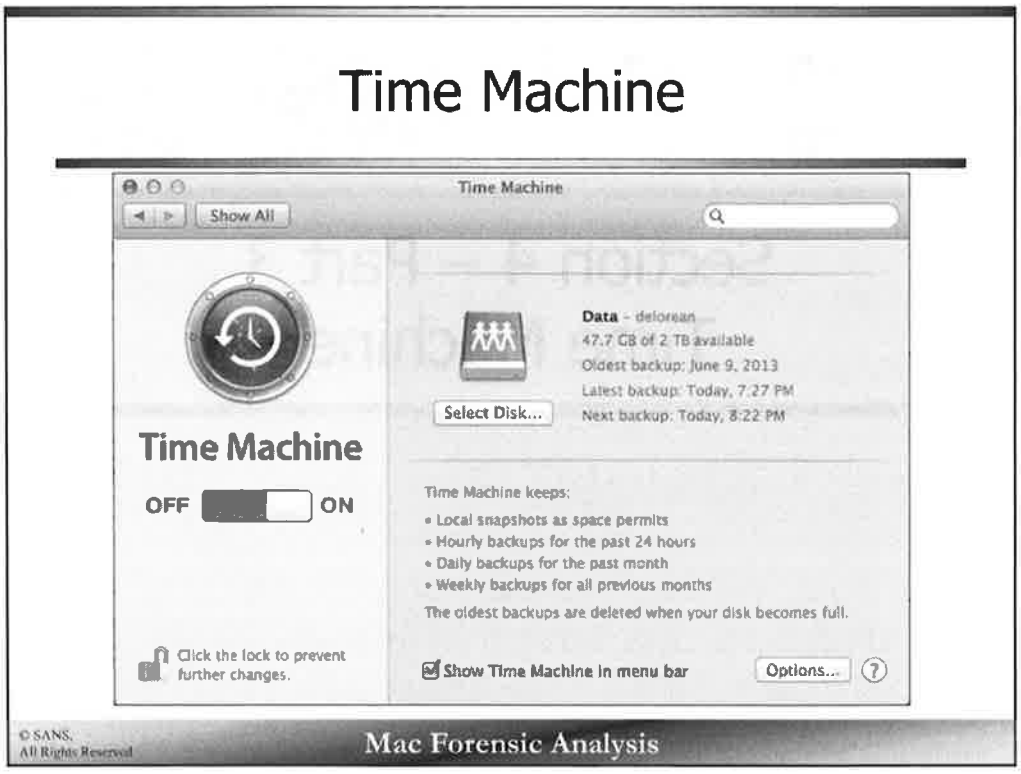
```
Users/sledwards/Downloads/hfsexplorer-0_21-bin/doc/html/index.html
Users/sledwards/Downloads/.DS_Store
Users/sledwards/Downloads/SFT-1.1.6-WIN.zip
Users/sledwards/Downloads/idx_parser.exe
Users/sledwards/Downloads/.DS_Store
Users/sledwards/Downloads/Makefile
Users/sledwards/Downloads/README
Users/sledwards/Downloads/dd2vmdk-0.1.1/README
Users/sledwards/Downloads/dd2vmdk-0.1.1/dd2vmdk
Users/sledwards/Downloads/dd2vmdk.c
Users/sledwards/Downloads/dd2vmdk.h
Users/sledwards/Downloads/v-0.1.1
Users/sledwards/Downloads/v-0.1.1/Makefile
Users/sledwards/Downloads/v-0.1.1/README
Users/sledwards/Downloads/v-0.1.1/dd2vmdk.c
Users/sledwards/Downloads/v-0.1.1/dd2vmdk.h
Users/sledwards/Downloads/.DS_Store
Users/sledwards/Downloads/PhoneView/.DS_Store
Users/sledwards/Downloads/PhoneView/PhoneView.app
Users/sledwards/Downloads/PhoneView/.DS_Store
Users/sledwards/Downloads/jre-7u17-macosx-x64-1.dmg
Users/sledwards/Downloads/jre-7u17-macosx-x64.dmg.download/Info.plist
Users/sledwards/Downloads/jre-7u17-macosx-x64.dmg.download/jre-7u17-macosx-x64-1.dmg
Users/sledwards/Downloads/meeting (1).jnlp
Users/sledwards/Downloads/.DS_Store
Users/sledwards/Downloads/photo.JPG
Users/sledwards/Downloads/photo.JPG
```



This page intentionally left blank.



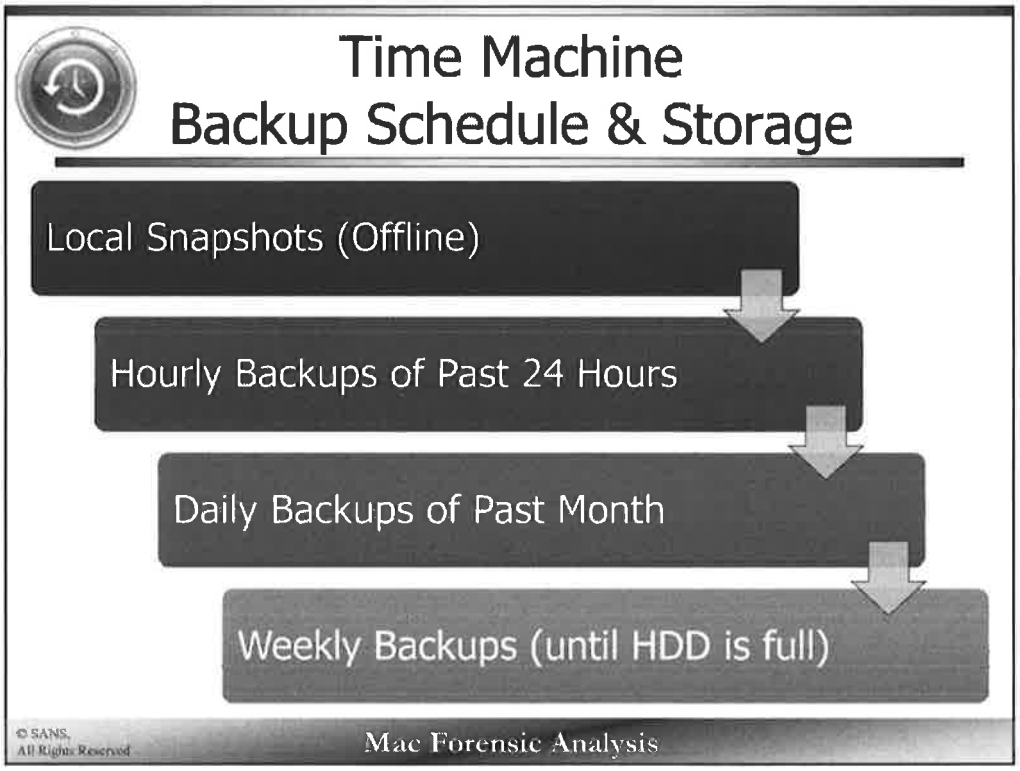
This page intentionally left blank.



The native backup program on OS X is called Time Machine, which was introduced in version 10.5. This application can interface with Apple Time Capsule, other network-based storage, or external hard drives.

Whenever the system detects a large external hard drive connected, or accesses a network storage device, a popup window will ask the user if they would like to use the disk as a Time Machine backup disk. These disks can be used with automated or manual backup systems depending on what configurations the user selects.

Shown in the screenshot above, a networked backup device named "`delorean`" contains a backup partition named "`Data`". This disk is two terabytes in size and has 47.7 gigabytes available. The Time Machine window will show data with respect to when the oldest and latest backups were completed and when the next backup is scheduled to take place.



Time Machine creates incremental backups of a system on a default schedule. This schedule can be changed by using the `defaults` command or editing the `com.apple.backupd.plist` property list file.

Local Snapshots, also called Mobile Backups, are created when the Time Machine volume is not present on the system… such as when a user is travelling and their Time Capsule is at home.

Hourly backups are created and kept for twenty-four hours. Daily backups are created and kept every day for the past month, and weekly backups are created and saved until the disk runs out of storage space.

Power Nap is a feature introduced in 10.8 for specific hardware systems (newer MacBook Pros and Airs). This allows the system to continue to create backups amongst other system functions such as checking for mail and downloading software updates while the Mac is in sleep mode.

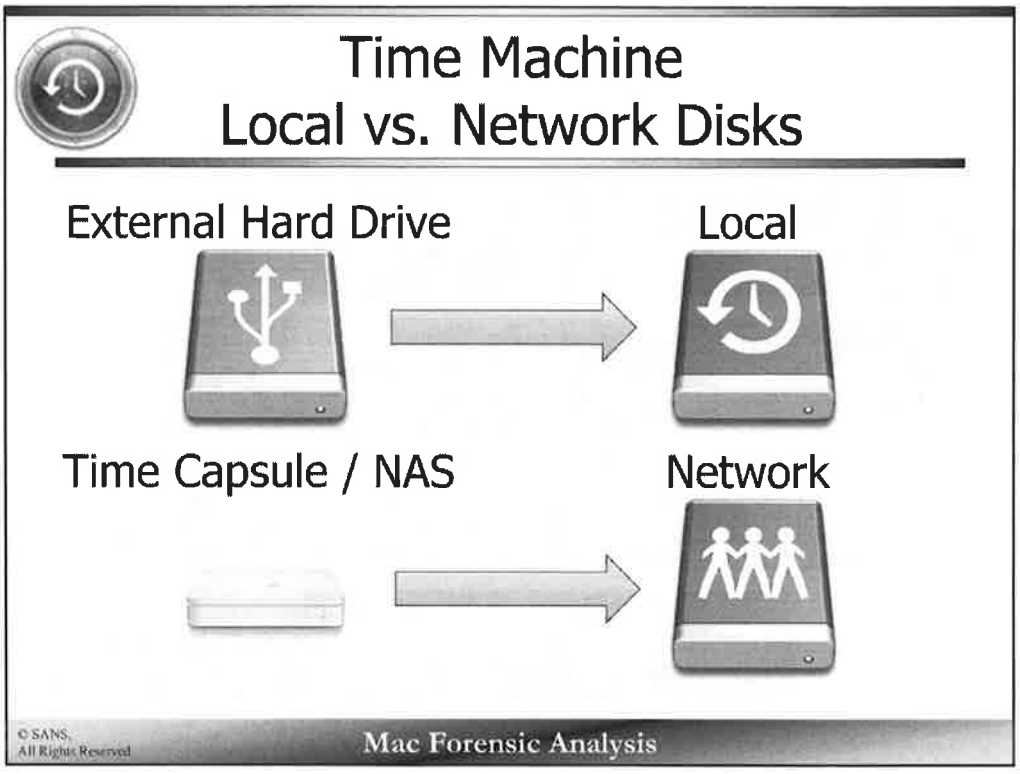
References:

Mac Basics – Time Machine

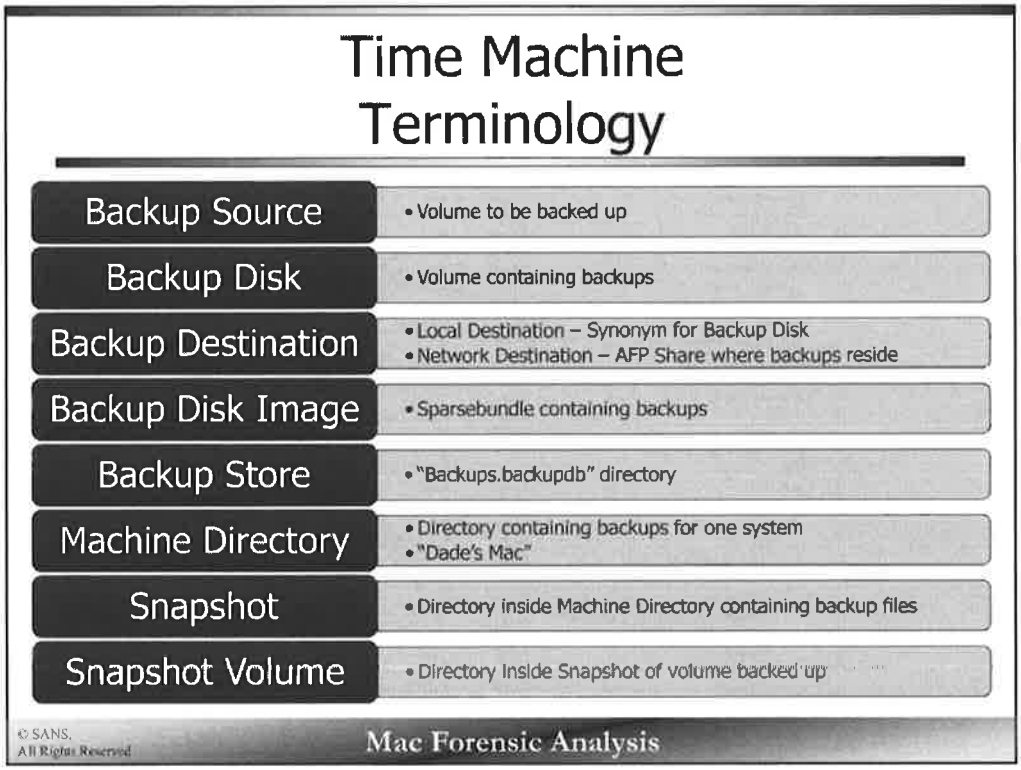
http://support.apple.com/kb/ht1427

Mountain Lion – Power Nap

http://support.apple.com/kb/ht5394



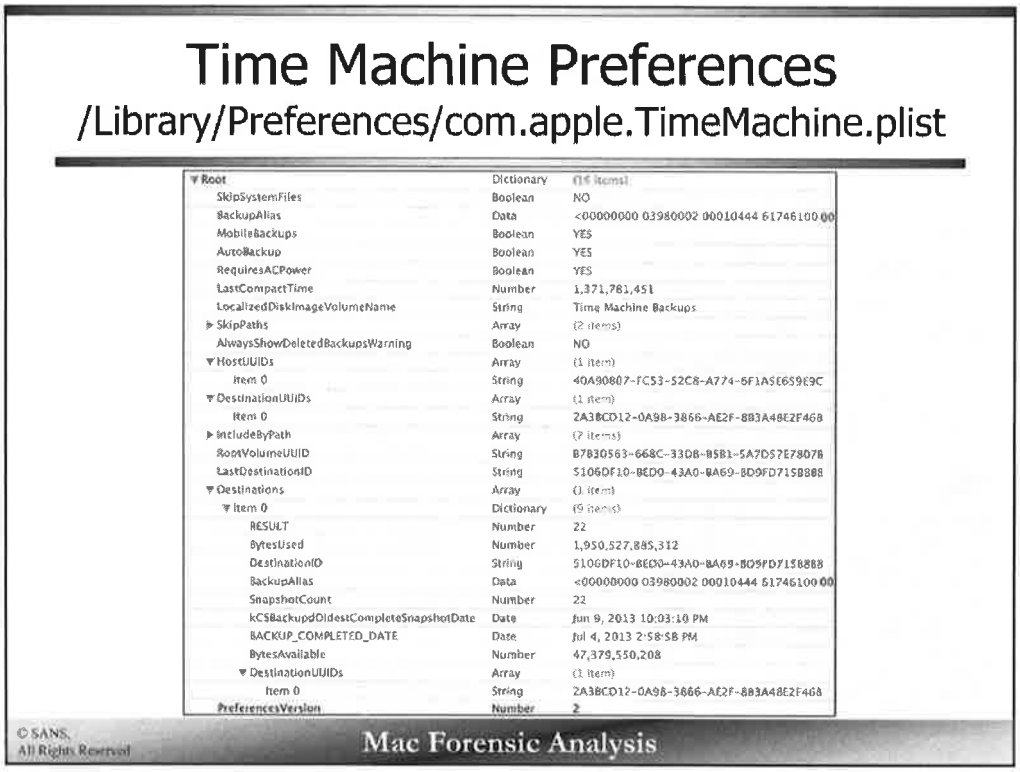
A network backup device has an icon that looks like a blue disk with paper dolls. An external (non-network) device will have an icon that looks like a green disk with a "Time Machine" arrow/clock – similar to the Time Machine application icon show above.



The Time Machine application uses specific terms to identify certain parts of the backup disk and process.

References:

`tmutil` Man Page



The backup data location used with Time Machine is located in the `com.apple.TimeMachine.plist` property list. This property list contains the GUIDs for the Destination Volume (the Time Capsule or external HDD) and the Root Volume (likely named Macintosh HD, the boot volume of the system). The `hostUUID` contains the GUID of the destination volume. These will all be explained a bit later in more detail.

This property list also contains certain Time Machine preference keys such as:

- `SkipSystemFiles` – If YES, the backup will skip system files
- `MobileBackups` – Enable backup creation while Time Machine volume is offline
- `AutoBackup` – Backup automatically (versus manual backups)
- `RequiresACPower` – Require backups to be performed while on AC power

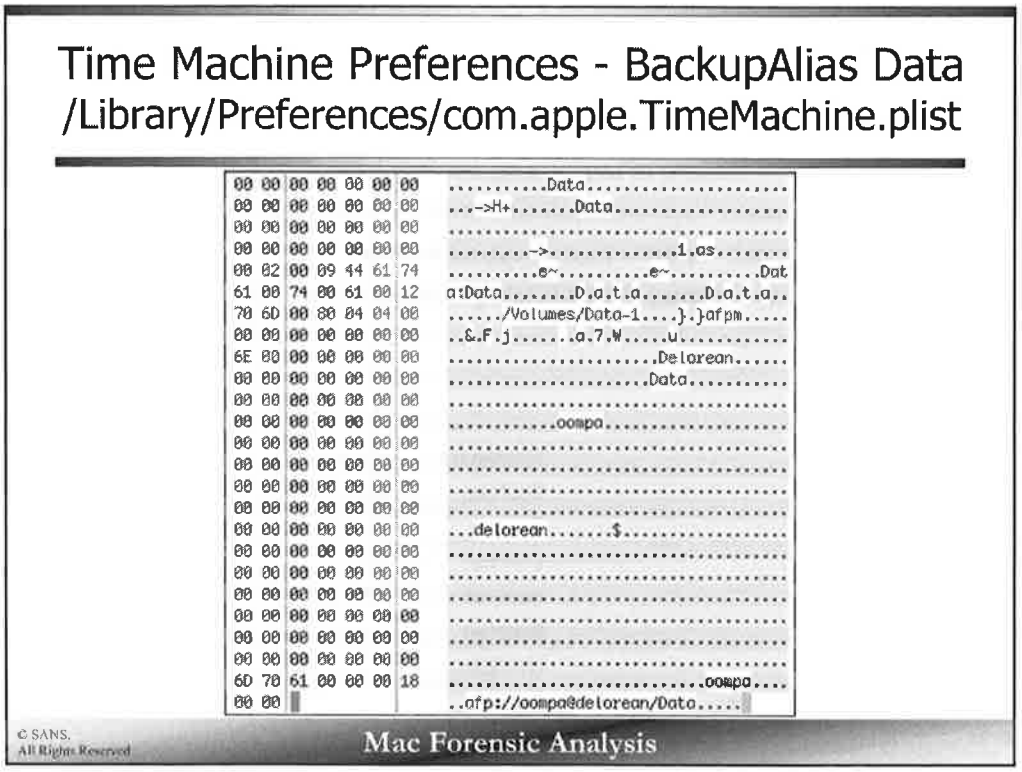
Other keys of interest include:

- `LastCompactTime` – The date in Unix epoch time that the backup sparse bundle has been "compacted" this creates more space. The sparse bundle is unique to network-based time machine volumes.
- `SkipPaths` – List of user configured file paths to explicitly skip on backup
- `IncludeByPath` – List of user configured file paths to explicitly include on backup
- `BackupAlias` – Binary alias data of the time machine backup volume

The `Destinations` key contains sub-keys for each time machine backup volume. These keys contain information pertaining to the backup volume such as:

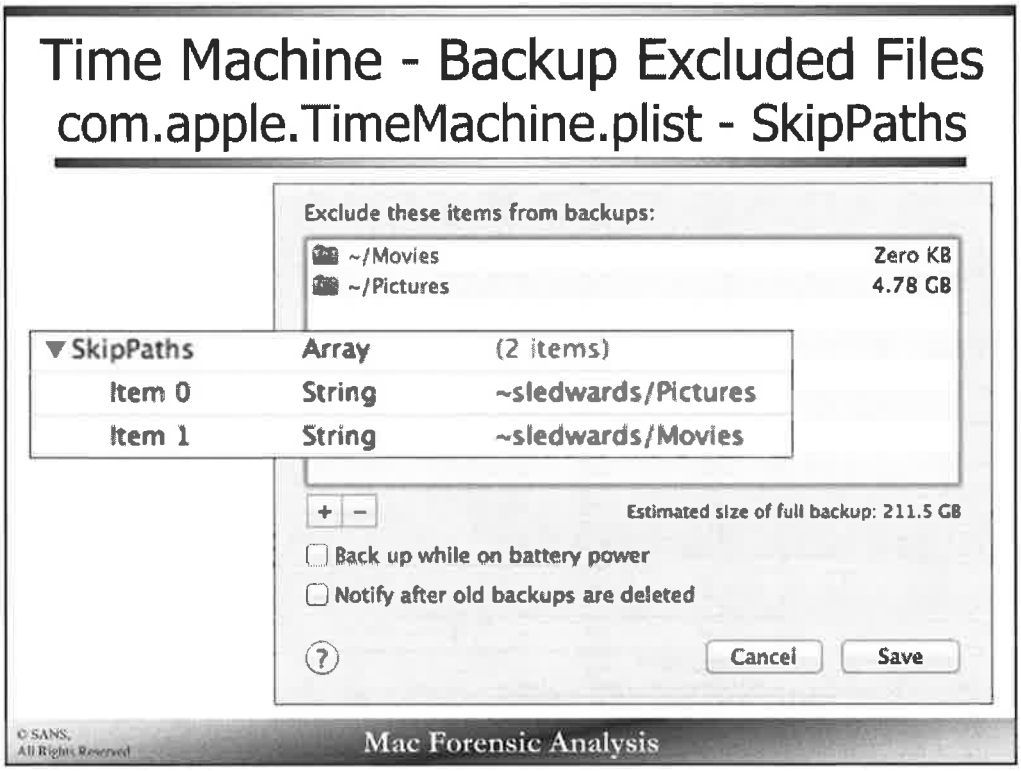
- `RESULT` – Result Code
- `BytesUsed` – Bytes allocated by the backups
- `SnapshotCount` – Number of backup snapshots available
- `kCSBackupdOldestCompleteSnapshotDate` – Oldest snapshot
- `BACKUP_COMPLETED_DATE` – Last completed backup
- `BytesAvailable` – Space available of the backup volume

▼ Root	Dictionary	(16 items)
SkipSystemFiles	Boolean	NO
BackupAlias	Data	<00000000 03980002 00010444 61746100 00
MobileBackups	Boolean	YES
AutoBackup	Boolean	YES
RequiresACPower	Boolean	YES
LastCompactTime	Number	1,371,781,451
LocalizedDiskImageVolumeName	String	Time Machine Backups
▶ SkipPaths	Array	(2 items)
AlwaysShowDeletedBackupsWarning	Boolean	NO
▼ HostUUIDs	Array	(1 item)
Item 0	String	40A90807-FC53-52C8-A774-6F1A5E659E9C
▼ DestinationUUIDs	Array	(1 item)
Item 0	String	2A3BCD12-0A98-3866-AE2F-8B3A48E2F468
▶ IncludeByPath	Array	(7 items)
RootVolumeUUID	String	87B30563-668C-33D8-B5B1-5A7D57E7807B
LastDestinationID	String	5106DF10-BED0-43A0-BA69-BD9FD715B8B8
▼ Destinations	Array	(1 item)
▼ Item 0	Dictionary	(9 items)
RESULT	Number	22
BytesUsed	Number	1,950,527,885,312
DestinationID	String	5106DF10-BED0-43A0-BA69-BD9FD715B8B8
BackupAlias	Data	<00000000 03980002 00010444 61746100 00
SnapshotCount	Number	22
kCSBackupdOldestCompleteSnapshotDate	Date	Jun 9, 2013 10:03:10 PM
BACKUP_COMPLETED_DATE	Date	Jul 4, 2013 2:58:58 PM
BytesAvailable	Number	47,379,550,208
▼ DestinationUUIDs	Array	(1 item)
Item 0	String	2A3BCD12-0A98-3866-AE2F-8B3A48E2F468
PreferencesVersion	Number	2



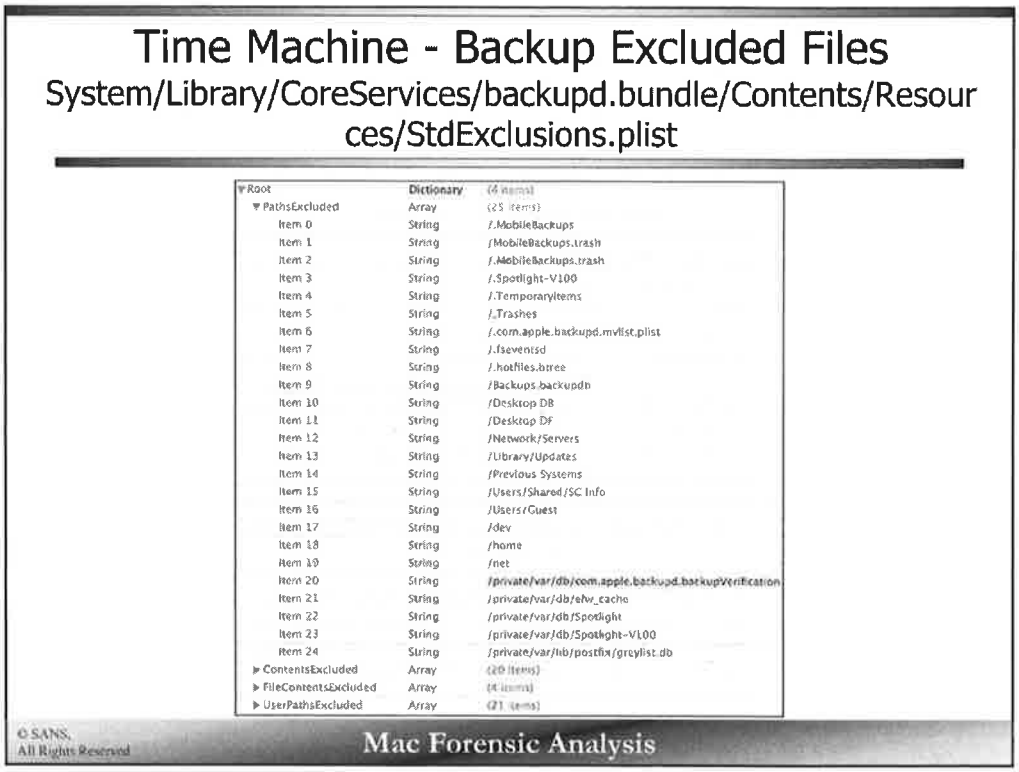
This data can be extracted and viewed in a hex editor. The key `BackupAlias` contains the backup volume location in binary data in the Alias form seen earlier.

The URL contains the handler "`afp://`", this protocol is the Apple Filing Protocol that is commonly used with Time Machine and the Apple network backup drive, Time Capsule. In the example we can see that the user creating the backup is "`oompa`". The Time Capsule is named "`delorean`" and the backup data is stored on the "`Data`" partition of the Time Capsule.



The `com.apple.TimeMachine.plist` file located in the `/Library/Preferences/` directory contains the key, `SkipPaths`. This key contains the directory file paths that will not be backed up

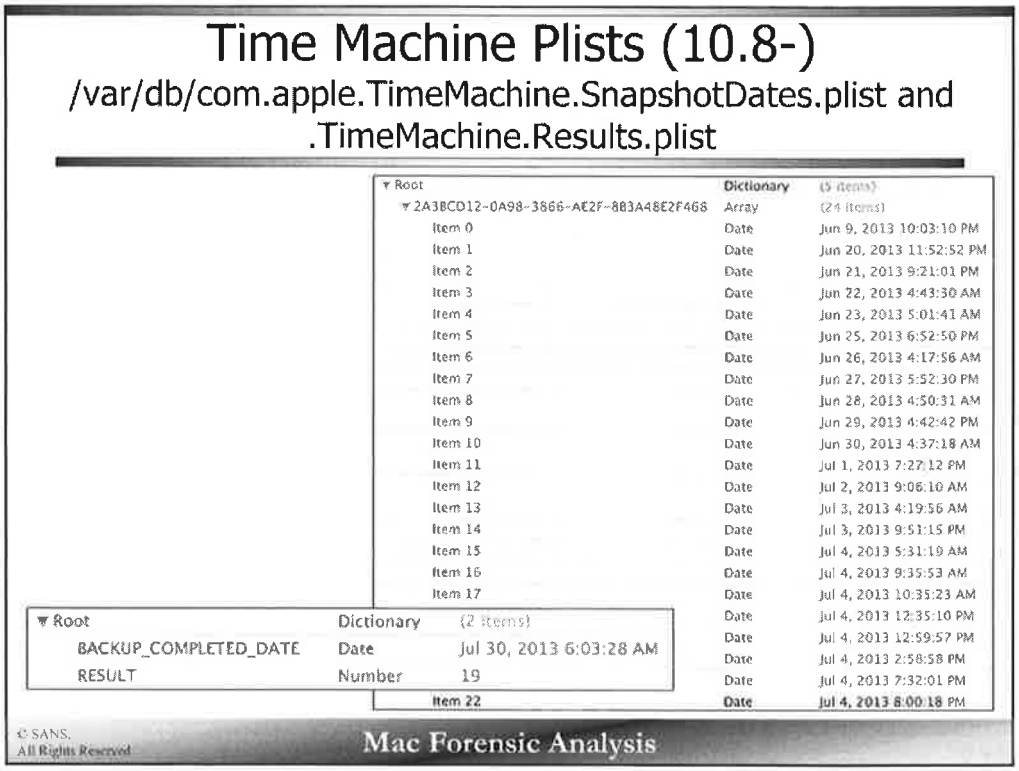
The larger screenshot shows what these paths look like in the Time Machine application GUI and in the smaller screenshot what they look like in the property list.



The `StdExclusions.plist` property list file located in the `/System/Library/CoreServices/backupd.bundle/Contents/Resources/` directory contains the default Time Machine exclusions.

This list contains the directory file paths for items that should also be excluded in Time Machine backups.

▼ Root	Dictionary	(4 items)
▼ PathsExcluded	Array	(25 items)
Item 0	String	/.MobileBackups
Item 1	String	/MobileBackups.trash
Item 2	String	/.MobileBackups.trash
Item 3	String	/.Spotlight-V100
Item 4	String	/.TemporaryItems
Item 5	String	/.Trashes
Item 6	String	/.com.apple.backupd.mvlist.plist
Item 7	String	/.fseventsd
Item 8	String	/.hotfiles.btree
Item 9	String	/Backups.backupdb
Item 10	String	/Desktop DB
Item 11	String	/Desktop DF
Item 12	String	/Network/Servers
Item 13	String	/Library/Updates
Item 14	String	/Previous Systems
Item 15	String	/Users/Shared/SC Info
Item 16	String	/Users/Guest
Item 17	String	/dev
Item 18	String	/home
Item 19	String	/net
Item 20	String	/private/var/db/com.apple.backupd.backupVerification
Item 21	String	/private/var/db/efw_cache
Item 22	String	/private/var/db/Spotlight
Item 23	String	/private/var/db/Spotlight-V100
Item 24	String	/private/var/lib/postfix/greylist.db
▶ ContentsExcluded	Array	(20 items)
▶ FileContentsExcluded	Array	(4 items)
▶ UserPathsExcluded	Array	(21 items)



The `com.apple.timemachine.snapshotdates.plist` property list located (shown in the larger screenshot) in the `/var/db/` directory contains the backups for Time Machine. Depending on the size and frequency of backups they can go all the way back to when the system was first configured.

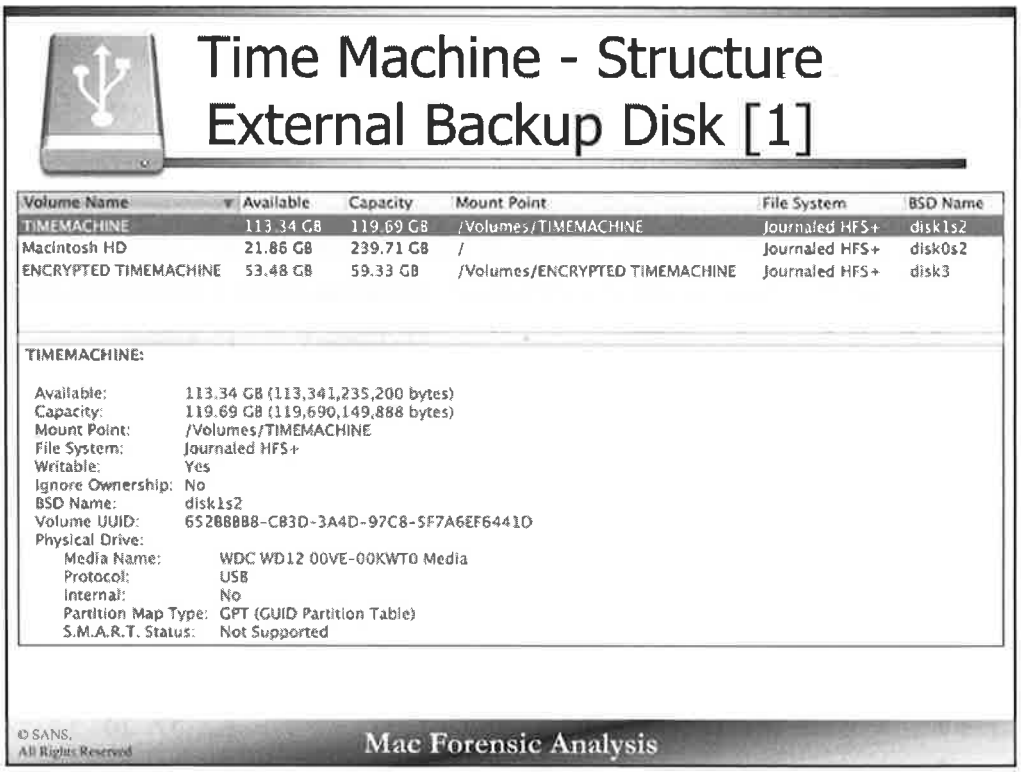
This property list contains the dates and times of each backup under the Destination UUID of the backup volume, in the example above - `2A3BCD12-0A98-3866-AE2F-8B3A48E2F468`.

This file may contain multiple Destination UUID if multiple backup volumes were used on the system.

The hidden file `.TimeMachine.Results.plist (shown in the smaller screenshot)` contains the timestamp of the last Time Machine Backup in the key `BACKUP_COMPLETED_DATE`, with the result code. This file is located in the `/private/var/db/` directory. It is unknown at this time what the results codes indicate.

These property lists are no longer found on 10.9 systems. On newer systems, this Snapshot Dates information is stored in the /Library/Preferences/com.apple.TimeMachine.plist

▼ Root	Dictionary	(5 items)
▼ 2A3BCD12-0A98-3866-AE2F-8B3A48E2F468	Array	(24 items)
Item 0	Date	Jun 9, 2013 10:03:10 PM
Item 1	Date	Jun 20, 2013 11:52:52 PM
Item 2	Date	Jun 21, 2013 9:21:01 PM
Item 3	Date	Jun 22, 2013 4:43:30 AM
Item 4	Date	Jun 23, 2013 5:01:41 AM
Item 5	Date	Jun 25, 2013 6:52:50 PM
Item 6	Date	Jun 26, 2013 4:17:56 AM
Item 7	Date	Jun 27, 2013 5:52:30 PM
Item 8	Date	Jun 28, 2013 4:50:31 AM
Item 9	Date	Jun 29, 2013 4:42:42 PM
Item 10	Date	Jun 30, 2013 4:37:18 AM
Item 11	Date	Jul 1, 2013 7:27:12 PM
Item 12	Date	Jul 2, 2013 9:06:10 AM
Item 13	Date	Jul 3, 2013 4:19:56 AM
Item 14	Date	Jul 3, 2013 9:51:15 PM
Item 15	Date	Jul 4, 2013 5:31:19 AM
Item 16	Date	Jul 4, 2013 9:35:53 AM
Item 17	Date	Jul 4, 2013 10:35:23 AM
Item 18	Date	Jul 4, 2013 12:35:10 PM
Item 19	Date	Jul 4, 2013 12:59:57 PM
Item 20	Date	Jul 4, 2013 2:58:58 PM
Item 21	Date	Jul 4, 2013 7:32:01 PM
Item 22	Date	Jul 4, 2013 8:00:18 PM



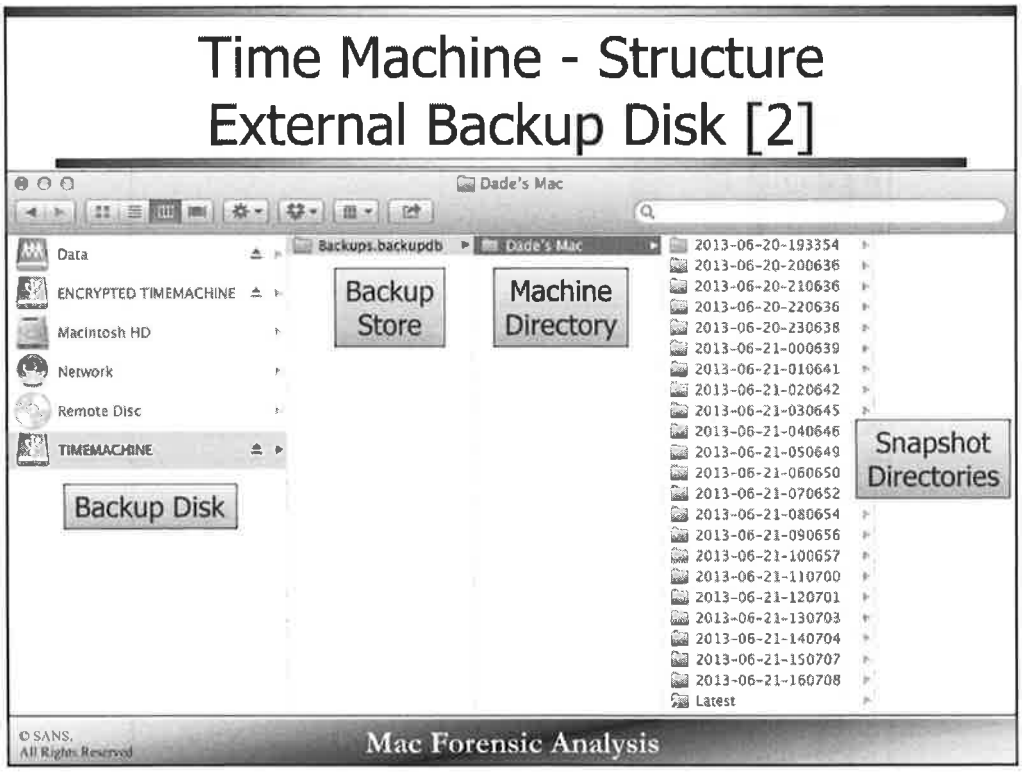
The TIMEMACHINE volume is shown in the screenshot above using the System Information application. This volume is using the HFS+ (Journaled) file system. All Time Machine volumes require the use of HFS+. It is not possible to have a Time Machine backup on FAT or NTFS formatted drives.

The screenshot also shows us the volume UUID, disk size, and connection protocol (USB).

Volume Name	Available	Capacity	Mount Point	File System	BSD Name
TIMEMACHINE	113.34 GB	119.69 GB	/Volumes/TIMEMACHINE	Journaled HFS+	disk1s2
Macintosh HD	21.86 GB	239.71 GB	/	Journaled HFS+	disk0s2
ENCRYPTED TIMEMACHINE	53.48 GB	59.33 GB	/Volumes/ENCRYPTED TIMEMACHINE	Journaled HFS+	disk3

TIMEMACHINE:

Available:	113.34 GB (113,341,235,200 bytes)
Capacity:	119.69 GB (119,690,149,888 bytes)
Mount Point:	/Volumes/TIMEMACHINE
File System:	Journaled HFS+
Writable:	Yes
Ignore Ownership:	No
BSD Name:	disk1s2
Volume UUID:	652BBBB8-CB3D-3A4D-97C8-5F7A6EF6441D
Physical Drive:	
Media Name:	WDC WD12 00VE-00KWT0 Media
Protocol:	USB
Internal:	No
Partition Map Type:	GPT (GUID Partition Table)
S.M.A.R.T. Status:	Not Supported

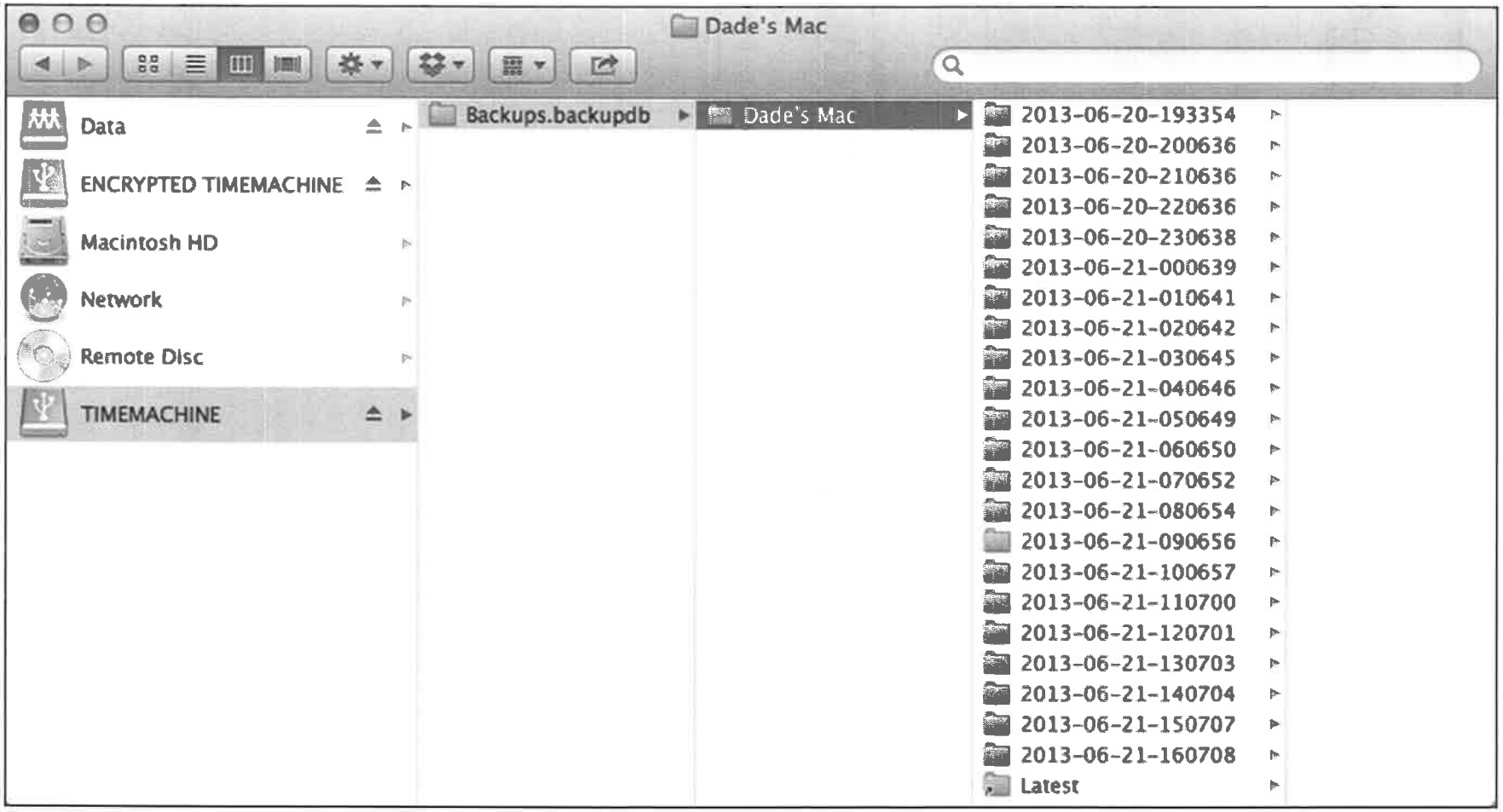


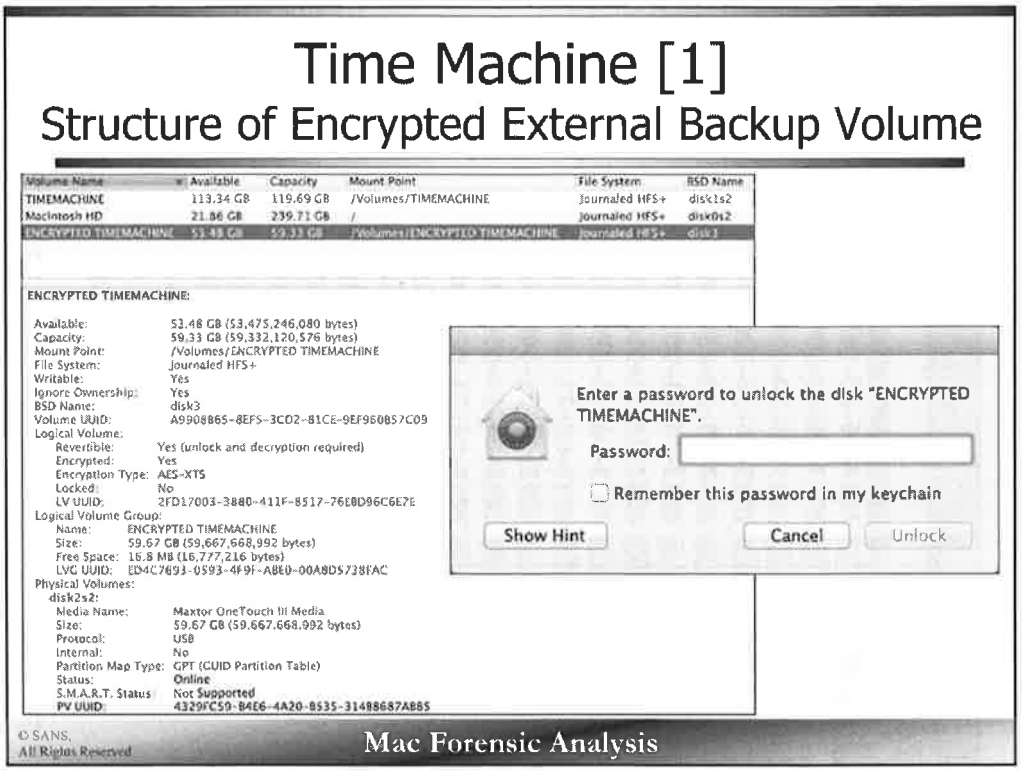
Using Finder we can look at the Time Machine related directories on the `TIMEMACHINE` volume:

- Backup Store - `Backups.backupdb/`
- Machine Directory - `Dade's Mac/`
- Snapshot directories – Timestamped Directories

The time stamped snapshot directories follow the naming convention of YYYY-MM-DD-HHMMSS (in 24 hour local system time).

If more than one system is backed up to this Time Machine there will be multiple Machine Directories.


Dade's Mac
Data
ENCRYPTED TIMEMACHINE
Macintosh HD
Network
Remote Disc
TIMEMACHINE
Backups.backupdb
Dade's Mac
2013-06-20-193354
2013-06-20-200636
2013-06-20-210636
2013-06-20-220636
2013-06-20-230638
2013-06-21-000639
2013-06-21-010641
2013-06-21-020642
2013-06-21-030645
2013-06-21-040646
2013-06-21-050649
2013-06-21-060650
2013-06-21-070652
2013-06-21-080654
2013-06-21-090656
2013-06-21-100657
2013-06-21-110700
2013-06-21-120701
2013-06-21-130703
2013-06-21-140704
2013-06-21-150707
2013-06-21-160708
Latest

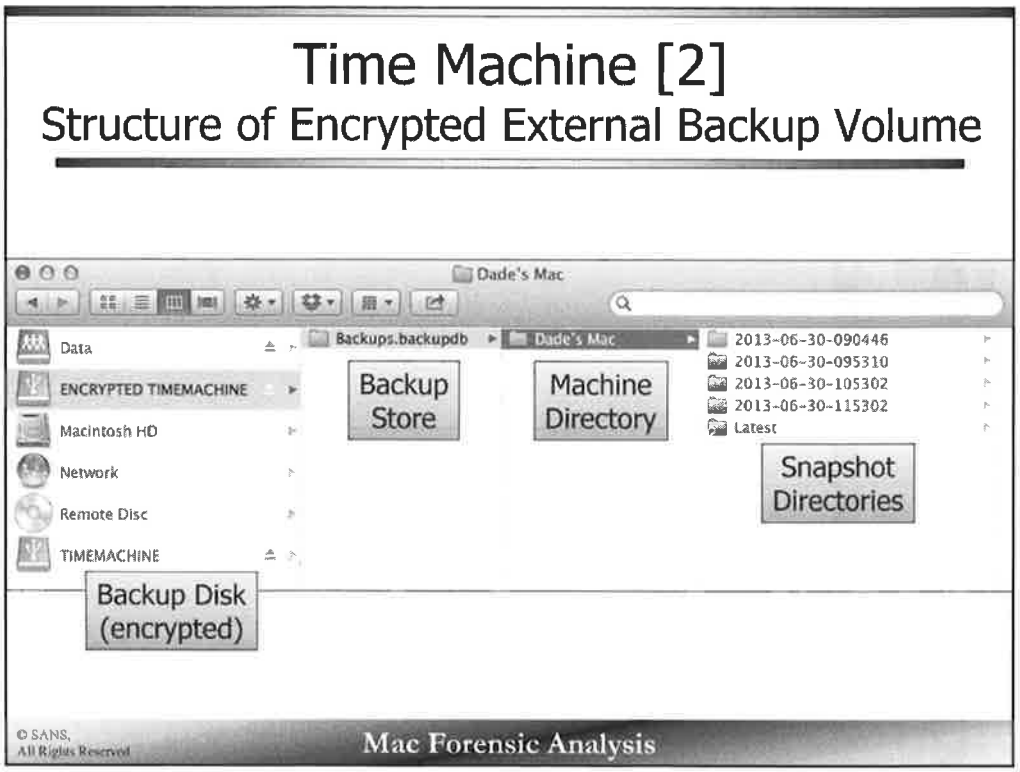


When an encrypted Time Machine is mounted (for example, from a forensic image) it will ask for a password as shown in the screenshot above. This password allows access to mount the encrypted CoreStorage volume as shown in the larger screenshot above.

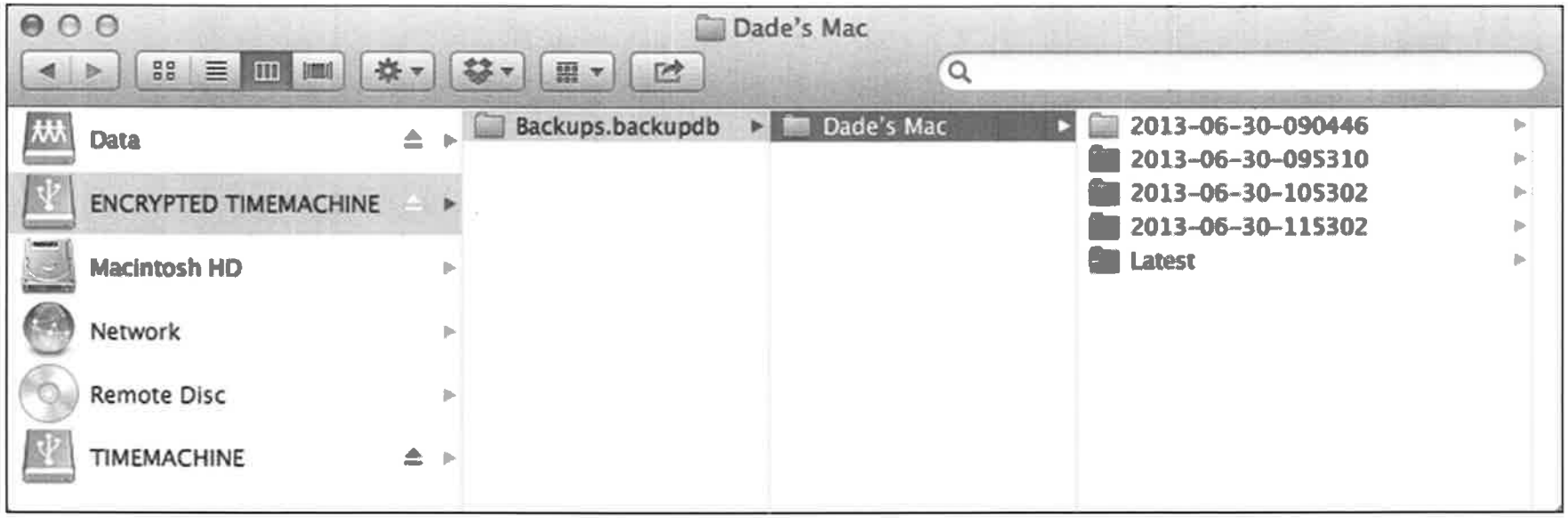
Volume Name	Available	Capacity	Mount Point	File System	BSD Name
TIMEMACHINE	113.34 GB	119.69 GB	/Volumes/TIMEMACHINE	Journaled HFS+	disk1s2
Macintosh HD	21.86 GB	239.71 GB	/	Journaled HFS+	disk0s2
ENCRYPTED TIMEMACHINE	53.48 GB	59.33 GB	/Volumes/ENCRYPTED TIMEMACHINE	Journaled HFS+	disk3

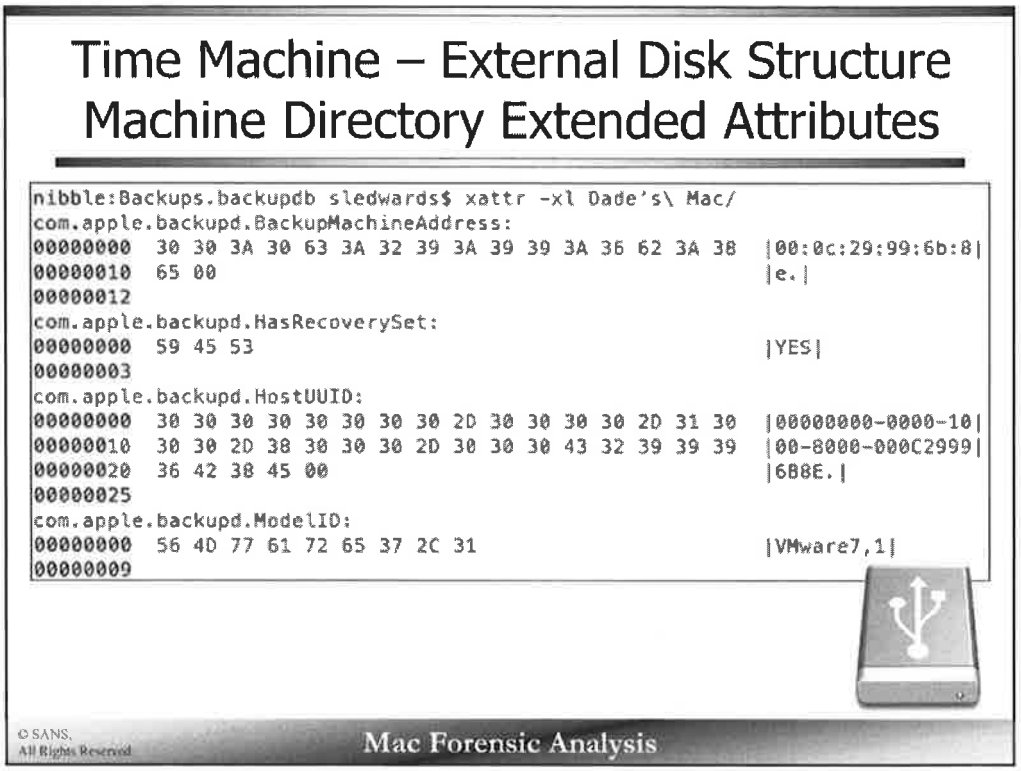
ENCRYPTED TIMEMACHINE:

Available: 53.48 GB (53,475,246,080 bytes)
Capacity: 59.33 GB (59,332,120,576 bytes)
Mount Point: /Volumes/ENCRYPTED TIMEMACHINE
File System: Journaled HFS+
Writable: Yes
Ignore Ownership: Yes
BSD Name: disk3
Volume UUID: A9908B65-8EF5-3CD2-81CE-9EF960B57C09
Logical Volume:
  Revertible: Yes (unlock and decryption required)
  Encrypted: Yes
  Encryption Type: AES-XTS
  Locked: No
  LV UUID: 2FD17003-3880-411F-8517-76EBD96C6E7E
Logical Volume Group:
  Name: ENCRYPTED TIMEMACHINE
  Size: 59.67 GB (59,667,668,992 bytes)
  Free Space: 16.8 MB (16,777,216 bytes)
  LVG UUID: ED4C7693-0593-4F9F-ABE0-00A8D5738FAC
Physical Volumes:
  disk2s2:
    Media Name: Maxtor OneTouch III Media
    Size: 59.67 GB (59,667,668,992 bytes)
    Protocol: USB
    Internal: No
    Partition Map Type: GPT (GUID Partition Table)
    Status: Online
    S.M.A.R.T. Status: Not Supported
    PV UUID: 4329FC59-B4E6-4A20-B535-314B8687AB85



Once mounted, the encrypted backup volume follows the same basic structure as a non-encrypted external backup volume, including metadata.


Dade's Mac
Data
ENCRYPTED TIMEMACHINE
Macintosh HD
Network
Remote Disc
TIMEMACHINE
Backups.backupdb
Dade's Mac
2013-06-30-090446
2013-06-30-095310
2013-06-30-105302
2013-06-30-115302
Latest



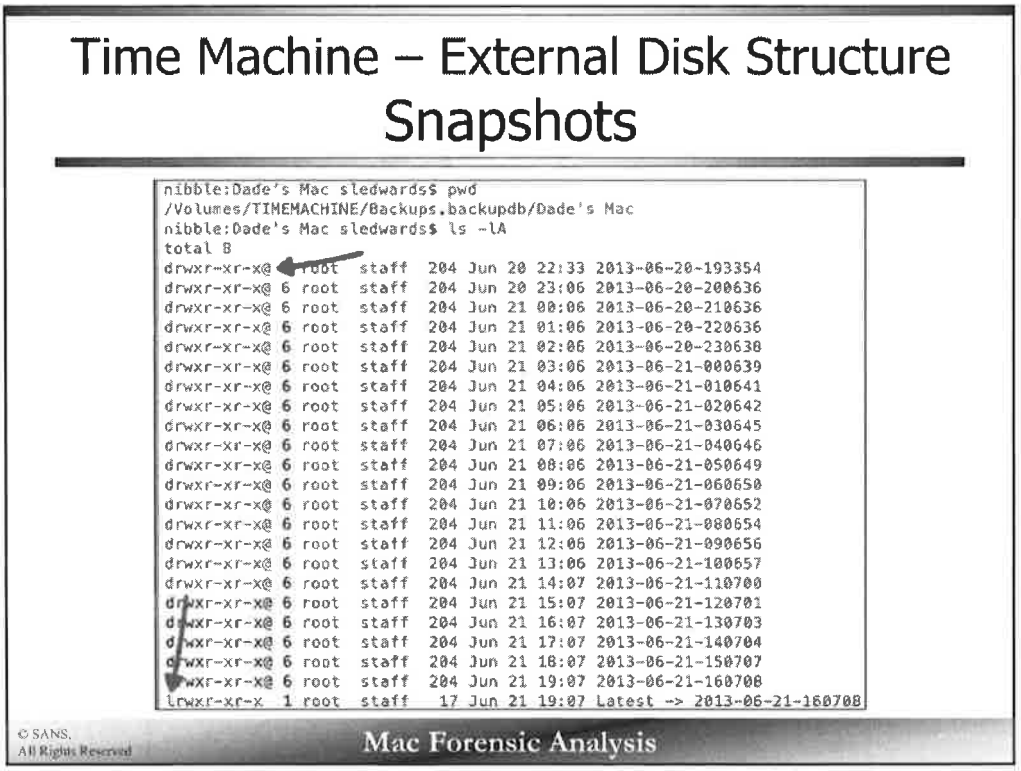
Using the extended attributes `xattr` command we can view the attributes for the Machine Directory '`Dade's Mac`'. The Machine Directory contains the following extended attributes:

- `com.apple.backupd.BackupMachineAddress` – Network MAC Address of the system
- `com.apple.backupd.HasRecoverySet` - Ability to boot from backup for recovery (look for the `Backups.backupdb/.RecoverySets` directory)
- `com.apple.backupd.HostUUID` – Hardware UUID
- `com.apple.backupd.ModelID` – Hardware Model Identifier

If the user selected "Encrypted Backups" you will find an additional extended attribute "`com.apple.backupd.HasEncryptedRecoveryBits`" set to "YES".

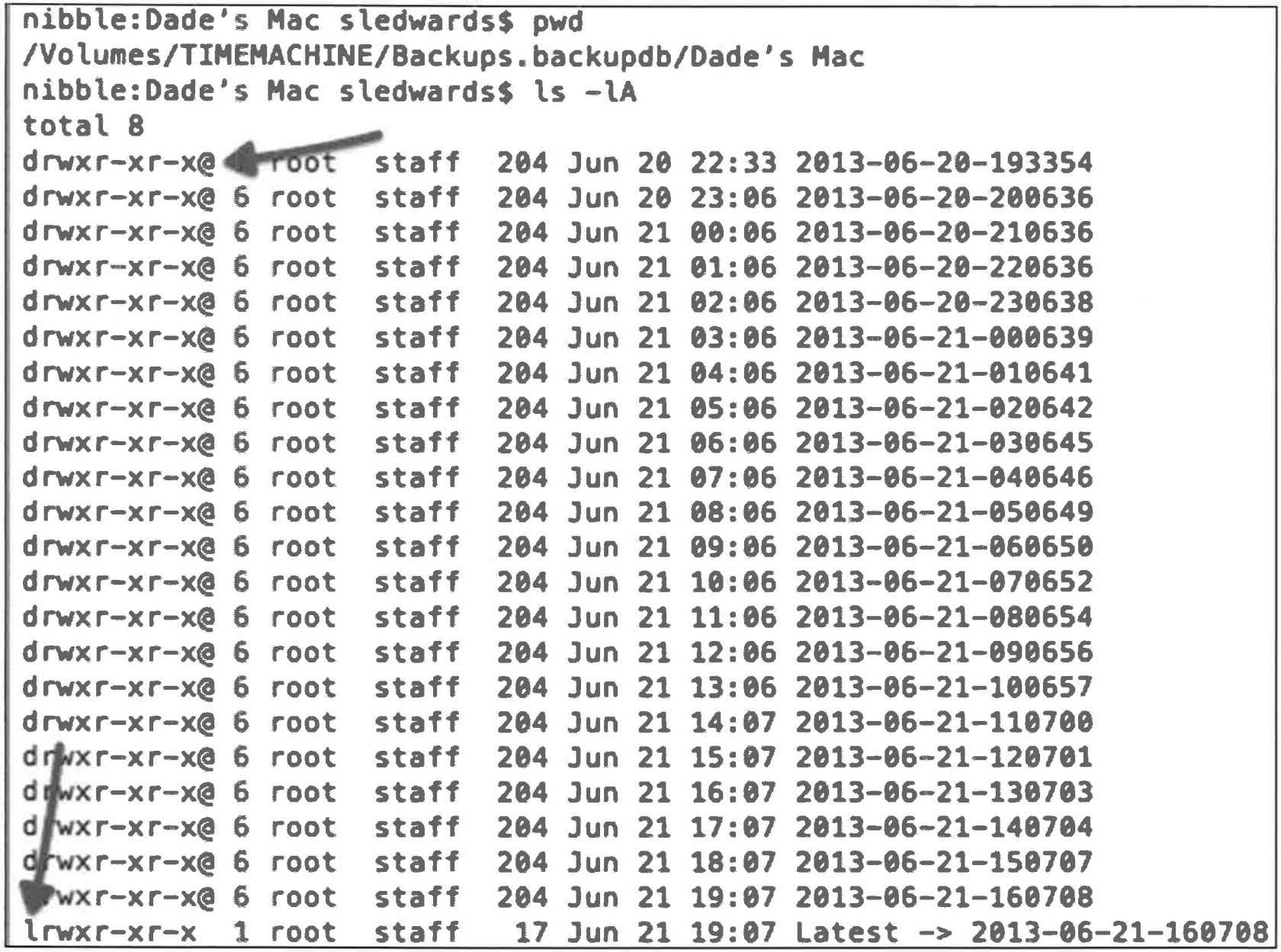
Investigators can use this information to determine the type of system these backups are from and potentially find other systems to analyze.

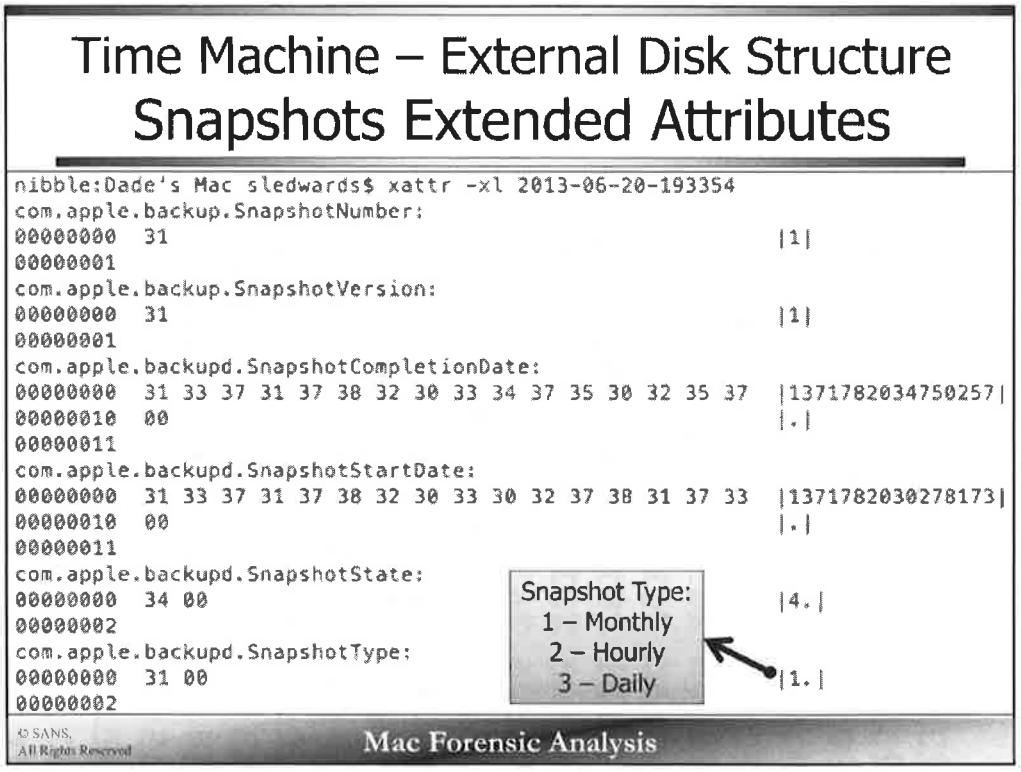
```
nibble:Backups.backupdb sledwards$ xattr -xl Dade's\ Mac/
com.apple.backupd.BackupMachineAddress:
00000000  30 30 3A 30 63 3A 32 39 3A 39 39 3A 36 62 3A 38  |00:0c:29:99:6b:8|
00000010  65 00                                            |e.|
00000012
com.apple.backupd.HasRecoverySet:
00000000  59 45 53                                         |YES|
00000003
com.apple.backupd.HostUUID:
00000000  30 30 30 30 30 30 30 30 2D 30 30 30 30 2D 31 30  |00000000-0000-10|
00000010  30 30 2D 38 30 30 30 2D 30 30 30 43 32 39 39 39  |00-8000-000C2999|
00000020  36 42 38 45 00                                   |6B8E.|
00000025
com.apple.backupd.ModelID:
00000000  56 4D 77 61 72 65 37 2C 31                       |VMware7,1|
00000009
```



Note the '1' in the metadata of the Latest Snapshot (it is located at the beginning of the attributes section) . This is a link pointing to the latest snapshot directory, `2013-06-21-160708`.

Each Snapshot directory contains its own extended attributes as indicated by the '@' in the directory listing.


nibble:Dade's Mac sledwards$ pwd
/Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac
nibble:Dade's Mac sledwards$ ls -lA
total 8
drwxr-xr-x@ 6 root staff 204 Jun 20 22:33 2013-06-20-193354
drwxr-xr-x@ 6 root staff 204 Jun 20 23:06 2013-06-20-200636
drwxr-xr-x@ 6 root staff 204 Jun 21 00:06 2013-06-20-210636
drwxr-xr-x@ 6 root staff 204 Jun 21 01:06 2013-06-20-220636
drwxr-xr-x@ 6 root staff 204 Jun 21 02:06 2013-06-20-230638
drwxr-xr-x@ 6 root staff 204 Jun 21 03:06 2013-06-21-000639
drwxr-xr-x@ 6 root staff 204 Jun 21 04:06 2013-06-21-010641
drwxr-xr-x@ 6 root staff 204 Jun 21 05:06 2013-06-21-020642
drwxr-xr-x@ 6 root staff 204 Jun 21 06:06 2013-06-21-030645
drwxr-xr-x@ 6 root staff 204 Jun 21 07:06 2013-06-21-040646
drwxr-xr-x@ 6 root staff 204 Jun 21 08:06 2013-06-21-050649
drwxr-xr-x@ 6 root staff 204 Jun 21 09:06 2013-06-21-060650
drwxr-xr-x@ 6 root staff 204 Jun 21 10:06 2013-06-21-070652
drwxr-xr-x@ 6 root staff 204 Jun 21 11:06 2013-06-21-080654
drwxr-xr-x@ 6 root staff 204 Jun 21 12:06 2013-06-21-090656
drwxr-xr-x@ 6 root staff 204 Jun 21 13:06 2013-06-21-100657
drwxr-xr-x@ 6 root staff 204 Jun 21 14:07 2013-06-21-110700
drwxr-xr-x@ 6 root staff 204 Jun 21 15:07 2013-06-21-120701
drwxr-xr-x@ 6 root staff 204 Jun 21 16:07 2013-06-21-130703
drwxr-xr-x@ 6 root staff 204 Jun 21 17:07 2013-06-21-140704
drwxr-xr-x@ 6 root staff 204 Jun 21 18:07 2013-06-21-150707
drwxr-xr-x@ 6 root staff 204 Jun 21 19:07 2013-06-21-160708
lrwxr-xr-x 1 root staff 17 Jun 21 19:07 Latest -> 2013-06-21-160708



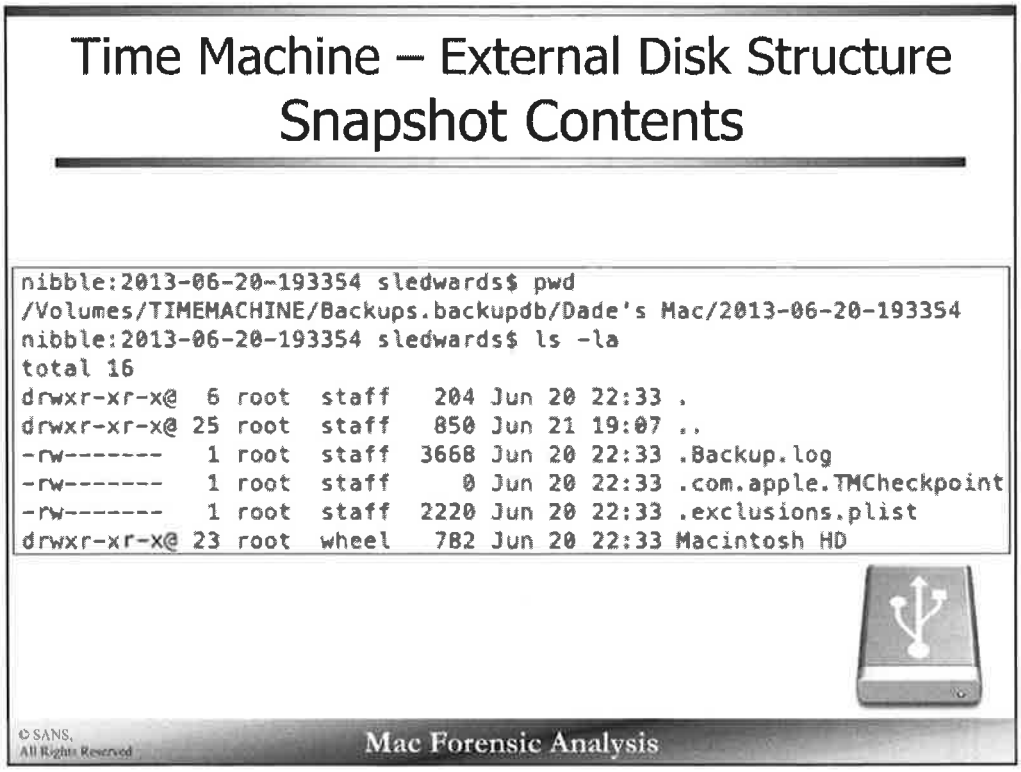
Using the extended attributes `xattr` command we can view the attributes for the Snapshot '`2013-06-20-193354`'. This is the initial backup for the example system. This snapshot contains the following extended attributes:

- `com.apple.backup.SnapshotNumber` – Incremental snapshot value (Note: It does not increment by single digits)
- `com.apple.backup.SnapshotVersion` – Appears to always be set to the value "1"
- `com.apple.backupd.SnapshotCompletionDate` – 16 digit Unix Epoch timestamp containing the backup completion date (input the first 10 digits into your favorite converter)
- `com.apple.backupd.SnapshotStartDate` - 16 digit Unix Epoch timestamp containing the start time of the backup
- `com.apple.backupd.SnapshotState` – Appears to always be set to the value "4" or `0x3400`
- `com.apple.backupd.SnapshotType` – Type of backup:
  - 1 – Monthly
  - 2 – Hourly
  - 3 – Daily

10.9 systems added an additional attribute "`com.apple.backupd.SnapshotTotalBytesCopied`" which contains the number of bytes copied into this snapshot.

Analysts can use this information to determine when a particular snapshot was created, and the type of snapshot it is.

```
nibble:Dade's Mac sledwards$ xattr -xl 2013-06-20-193354
com.apple.backup.SnapshotNumber:
00000000  31                                                |1|
00000001
com.apple.backup.SnapshotVersion:
00000000  31                                                |1|
00000001
com.apple.backupd.SnapshotCompletionDate:
00000000  31 33 37 31 37 38 32 30 33 34 37 35 30 32 35 37  |1371782034750257|
00000010  00                                                |.|
00000011
com.apple.backupd.SnapshotStartDate:
00000000  31 33 37 31 37 38 32 30 33 30 32 37 38 31 37 33  |1371782030278173|
00000010  00                                                |.|
00000011
com.apple.backupd.SnapshotState:
00000000  34 00                                             |4.|
00000002
com.apple.backupd.SnapshotType:
00000000  31 00                                             |1.|
00000002
```



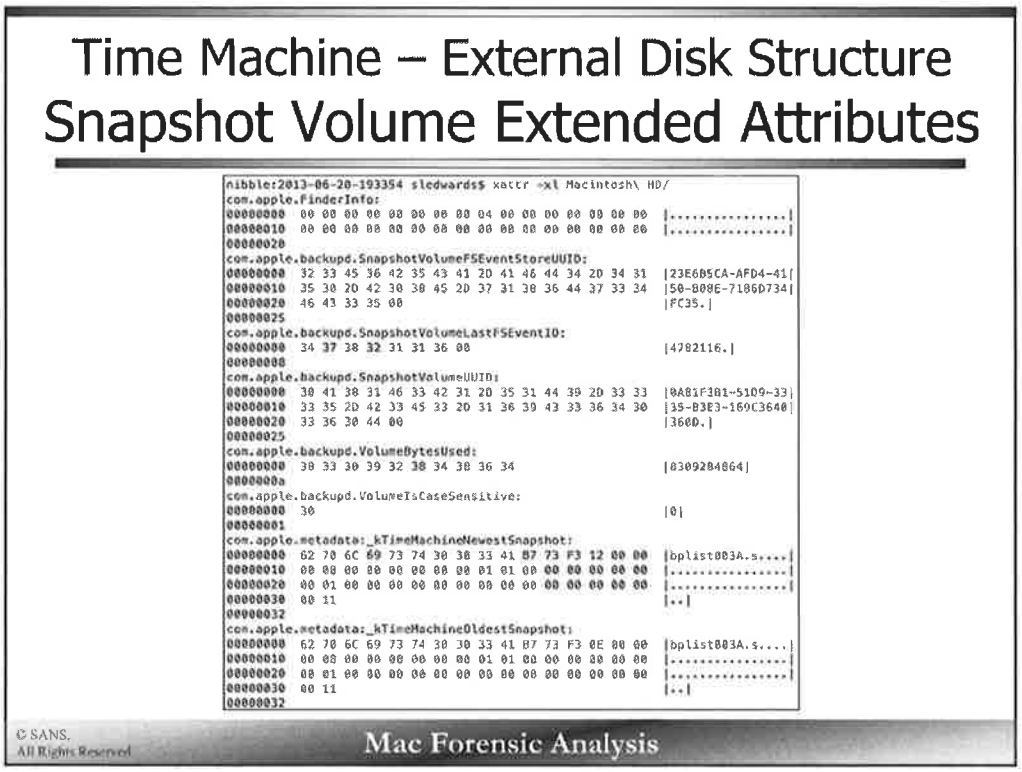
The screenshot shows the contents of the Snapshot `2013-06-20-193354`. The Snapshot volume name is '`Macintosh HD`'. This directory also contains the hidden files:

- `.Backup.log` – Contains the backup log
- `.com.apple.TMCheckpoint` – Unknown, size is always 0 bytes
- `.exclusions.plist` – Contains a list of the excluded files and directories for this snapshot

The `.Backup.log` file contains the backup log data for that specific snapshot. It will include items such as excluded directories, how much data was copied, and how long it took to create the snapshot. An example is shown on the following pages.

```
2013-06-20-19:05:56 - Starting backup

Previous snapshot:
            None

Will traverse "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-51D9-3335-B3E3-
169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-7186D734FC35)
=== Starting backup loop #1 ===
  Will use FirstBackupCopier

Running preflight for "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-51D9-3335-B3E3-
169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-7186D734FC35)
            Excluding /.Spotlight-V100: 50.6 MB (90 items)
            Excluding /.Trashes: Zero KB (1 items)
            Excluding /.fseventsd: 29 KB (8 items)
            Excluding /.hotfiles.btree: 66 KB (1 items)
            Excluding /Library/Updates: 1.03 GB (15 items)
            Excluding /private/var/db/Spotlight: Zero KB (2 items)
            Excluding /Volumes: 4 KB (4 items)
            Excluding /Network: Zero KB (1 items)
            Excluding /.vol: Zero KB (1 items)
            Excluding /cores: Zero KB (1 items)
            Excluding /private/tmp: Zero KB (7 items)
            Excluding /private/tftpboot: Zero KB (1 items)
            Excluding /private/var/folders: 504.4 MB (165 items)
            Excluding /private/var/run: 33 KB (15 items)
            Excluding /private/var/tmp: Zero KB (3 items)
            Excluding /private/var/vm: 67.1 MB (2 items)
            Excluding /private/var/db/dhcpclient: 4 KB (3 items)
            Excluding /Library/Caches: 12.7 MB (11 items)
            Excluding /Library/Logs: 4 KB (6 items)
            Excluding /System/Library/Caches: 22.1 MB (39 items)
            Excluding /private/var/log: 4.3 MB (92 items)
            Excluding /private/var/spool/cups: 8 KB (6 items)
            Excluding /private/var/spool/fax: Zero KB (1 items)
            Excluding /private/var/spool/uucp: Zero KB (1 items)
            Excluding /private/var/db/dyld: 523.6 MB (12 items)
            Should copy 326143 items (6.03 GB) representing 1471664 blocks of size
4096. 29166761 blocks available.
Preflight complete for "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-51D9-3335-B3E3-
169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-7186D734FC35)
Time elapsed: 0.246 seconds

Processing preflight info
            Space needed for this backup: 7.23 GB (1766133 blocks of size 4096)
Finished processing preflight info
```

```
Copying items from "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-51D9-3335-
B3E3-169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-7186D734FC35)
Finished copying items for "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-
51D9-3335-B3E3-169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-
7186D734FC35)
Time elapsed: 27 minutes, 52.000 seconds
            Copied 325141 items (5.56 GB)
Gathering events since 28327.
Needs new backup due to change in /private/var/db/.dat0207.001
=== Starting backup loop #2 ===
  Will use IncrementalBackupCopier

Running preflight for "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-51D9-
3335-B3E3-169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-7186D734FC35)
            Calculating size of changes
            Should copy 14 items (Zero KB) representing 0 blocks of size
4096. 27710787 blocks available.
Preflight complete for "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-51D9-
3335-B3E3-169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-7186D734FC35)
Time elapsed: 0.245 seconds

Processing preflight info
            Space needed for this backup: 402.5 MB (98257 blocks of size
4096)
            Preserving last snapshot
/Volumes/Untitled/Backups.backupdb/Dade's Mac/2013-06-20-
190556.inProgress/1F682AE6-31FF-42A5-9430-4D47C1BBB689
Finished processing preflight info

Copying items from "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-51D9-3335-
B3E3-169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-7186D734FC35)
Finished copying items for "Macintosh HD" (mount: '/' fsUUID: 0A81F3B1-
51D9-3335-B3E3-169C3640360D eventDBUUID: 23E6B5CA-AFD4-4150-B08E-
7186D734FC35)
Time elapsed: 4.122 seconds
            Copied 233 items (33 bytes)
Gathering events since 4782116.

Backup complete.
Total time elapsed: 27 minutes, 59.000 seconds
```

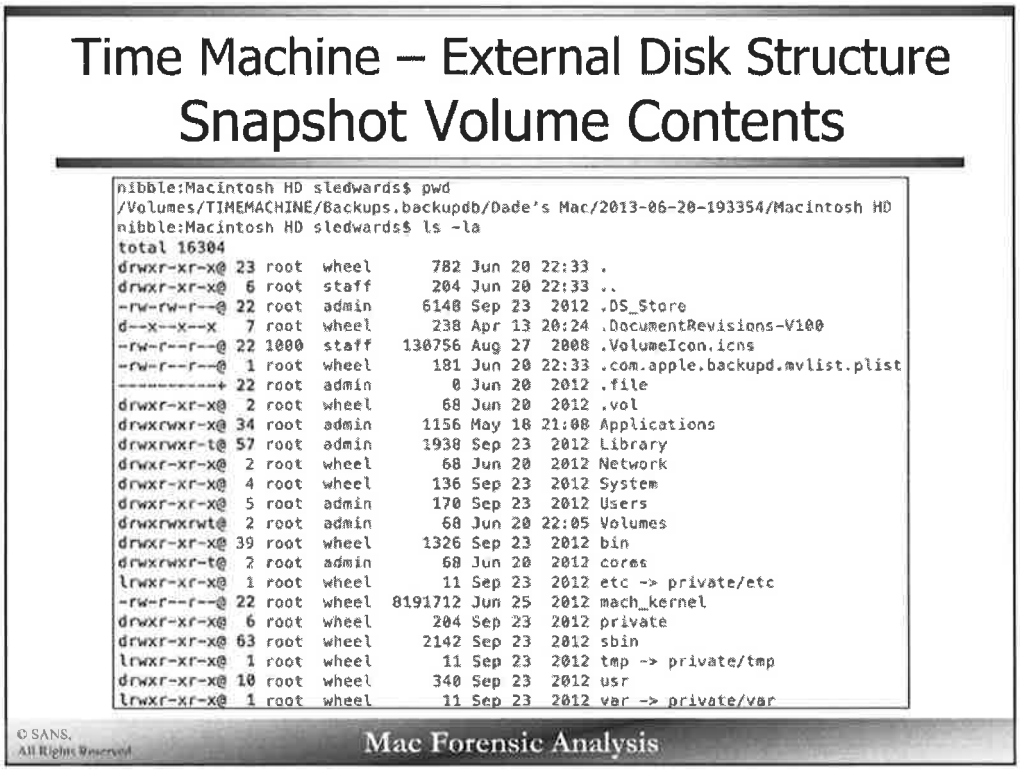


Using the extended attributes `xattr` command we can view the attributes for the Snapshot Volume 'Macintosh HD'.

- `com.apple.FinderInfo` – Finder Attributes
- `com.apple.backupd.SnapshotVolumeFSEventStoreUUID` – UUID of the Events Store
- `com.apple.backupd.SnapshotVolumeLastFSEventID` – Last Event Store ID
- `com.apple.backupd.SnapshotVolumeUUID` – UUID of this Snapshot Volume
- `com.apple.backupd.VolumeBytesUsed` – Snapshot volume size (if hard links are taken into account)
- `com.apple.backupd.VolumeIsCaseSensitive` – Case Sensitivity of Volume (0 = Not Case Sensitive)
- `com.apple.metadata:_kTimeMachineNewestSnapshot` – Binary plist containing the timestamp associated with the snapshot
- `com.apple.metadata:_kTimeMachineOldestSnapshot` – Binary plist containing the timestamp associated with the snapshot.

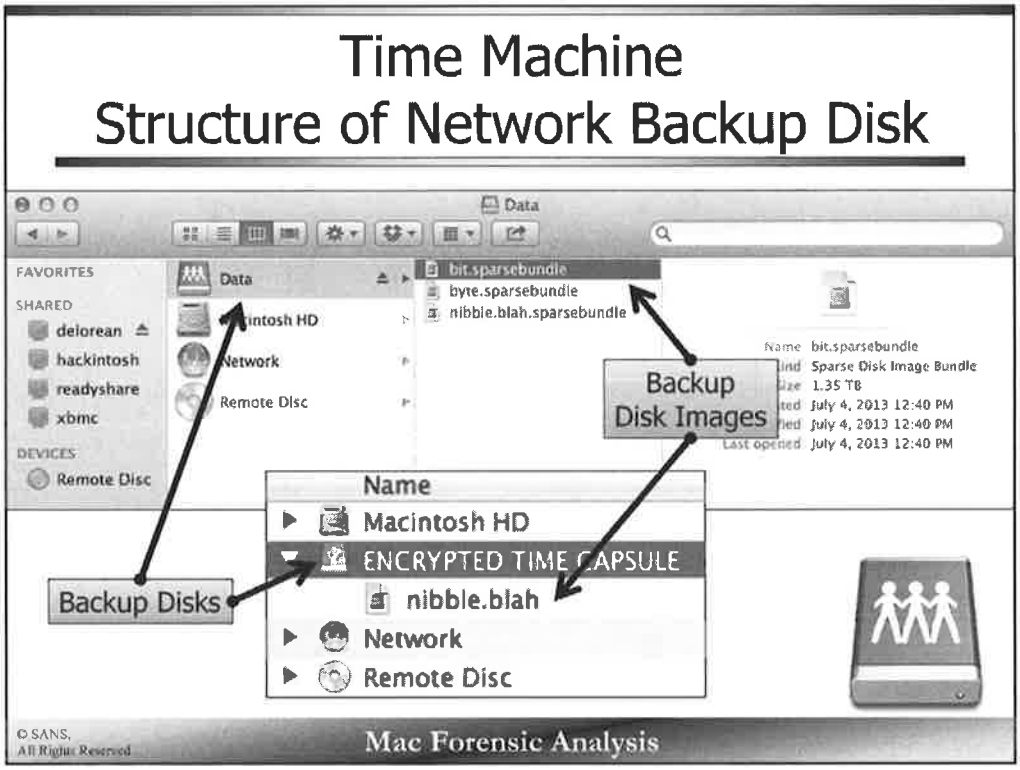
This information can be used to determine disk usage between snapshots. Knowing if a large amount of files were deleted or added to the backup may be important to an investigator.

```
nibble:2013-06-20-193354 sledwards$ xattr -xl Macintosh\ HD/
com.apple.FinderInfo:
00000000  00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000020
com.apple.backupd.SnapshotVolumeFSEventStoreUUID:
00000000  32 33 45 36 42 35 43 41 2D 41 46 44 34 2D 34 31  |23E6B5CA-AFD4-41|
00000010  35 30 2D 42 30 38 45 2D 37 31 38 36 44 37 33 34  |50-B08E-7186D734|
00000020  46 43 33 35 00                                    |FC35.|
00000025
com.apple.backupd.SnapshotVolumeLastFSEventID:
00000000  34 37 38 32 31 31 36 00                          |4782116.|
00000008
com.apple.backupd.SnapshotVolumeUUID:
00000000  30 41 38 31 46 33 42 31 2D 35 31 44 39 2D 33 33  |0A81F3B1-51D9-33|
00000010  33 35 2D 42 33 45 33 2D 31 36 39 43 33 36 34 30  |35-B3E3-169C3640|
00000020  33 36 30 44 00                                    |360D.|
00000025
com.apple.backupd.VolumeBytesUsed:
00000000  38 33 30 39 32 38 34 38 36 34                    |8309284864|
0000000a
com.apple.backupd.VolumeIsCaseSensitive:
00000000  30                                                |0|
00000001
com.apple.metadata:_kTimeMachineNewestSnapshot:
00000000  62 70 6C 69 73 74 30 30 33 41 B7 73 F3 12 00 00  |bplist003A.s....|
00000010  00 08 00 00 00 00 00 00 01 01 00 00 00 00 00 00  |................|
00000020  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 11                                             |..|
00000032
com.apple.metadata:_kTimeMachineOldestSnapshot:
00000000  62 70 6C 69 73 74 30 30 33 41 B7 73 F3 0E 00 00  |bplist003A.s....|
00000010  00 08 00 00 00 00 00 00 01 01 00 00 00 00 00 00  |................|
00000020  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 11                                             |..|
00000032
```



Looking into the Snapshot Volume "`Macintosh HD`" we can see a normal OS X file system. While it may look complete, behind the scenes there are many hard links to various files. This allows snapshots to link to all the data files while not having to have multiple copies of redundant data. These links point to the file in the snapshot that contains the original version of the file.

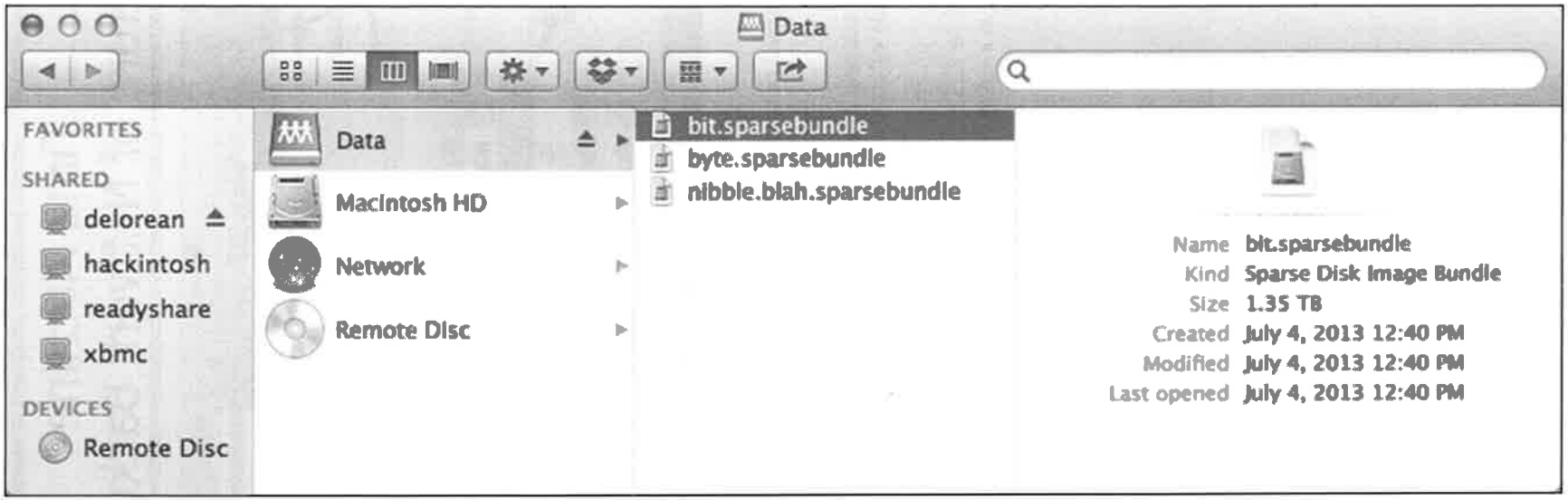
```
nibble:Macintosh HD sledwards$ pwd
/Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-193354/Macintosh HD
nibble:Macintosh HD sledwards$ ls -la
total 16304
drwxr-xr-x@ 23 root  wheel      782 Jun 20 22:33 .
drwxr-xr-x@  6 root  staff      204 Jun 20 22:33 ..
-rw-rw-r--@ 22 root  admin     6148 Sep 23  2012 .DS_Store
d--x--x--x   7 root  wheel      238 Apr 13 20:24 .DocumentRevisions-V100
-rw-r--r--@ 22 1000  staff   130756 Aug 27  2008 .VolumeIcon.icns
-rw-r--r--@  1 root  wheel      181 Jun 20 22:33 .com.apple.backupd.mvlist.plist
----------+ 22 root  admin        0 Jun 20  2012 .file
drwxr-xr-x@  2 root  wheel       68 Jun 20  2012 .vol
drwxrwxr-x@ 34 root  admin     1156 May 18 21:08 Applications
drwxrwxr-t@ 57 root  admin     1938 Sep 23  2012 Library
drwxr-xr-x@  2 root  wheel       68 Jun 20  2012 Network
drwxr-xr-x@  4 root  wheel      136 Sep 23  2012 System
drwxr-xr-x@  5 root  admin      170 Sep 23  2012 Users
drwxrwxrwt@  2 root  admin       68 Jun 20 22:05 Volumes
drwxr-xr-x@ 39 root  wheel     1326 Sep 23  2012 bin
drwxrwxr-t@  2 root  admin       68 Jun 20  2012 cores
lrwxr-xr-x@  1 root  wheel       11 Sep 23  2012 etc -> private/etc
-rw-r--r--@ 22 root  wheel  8191712 Jun 25  2012 mach_kernel
drwxr-xr-x@  6 root  wheel      204 Sep 23  2012 private
drwxr-xr-x@ 63 root  wheel     2142 Sep 23  2012 sbin
lrwxr-xr-x@  1 root  wheel       11 Sep 23  2012 tmp -> private/tmp
drwxr-xr-x@ 10 root  wheel      340 Sep 23  2012 usr
lrwxr-xr-x@  1 root  wheel       11 Sep 23  2012 var -> private/var
```

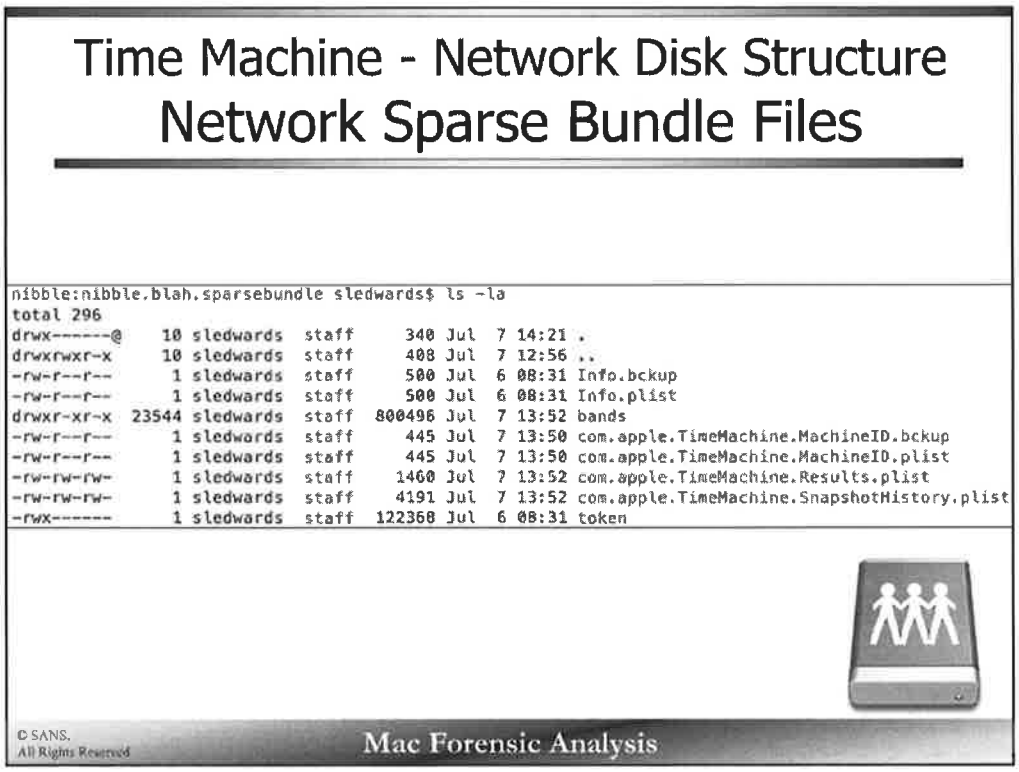


Both non-encrypted and encrypted network backup disks use the sparse bundle file format.

The top screenshot shows an example from an unencrypted disk, while the inset screenshot shows an example from an encrypted time capsule volume.

Note: While the icon in the lower screenshot shows a USB drive, this is an external hard drive that was connected to a Time Capsule as a secondary network hard drive.

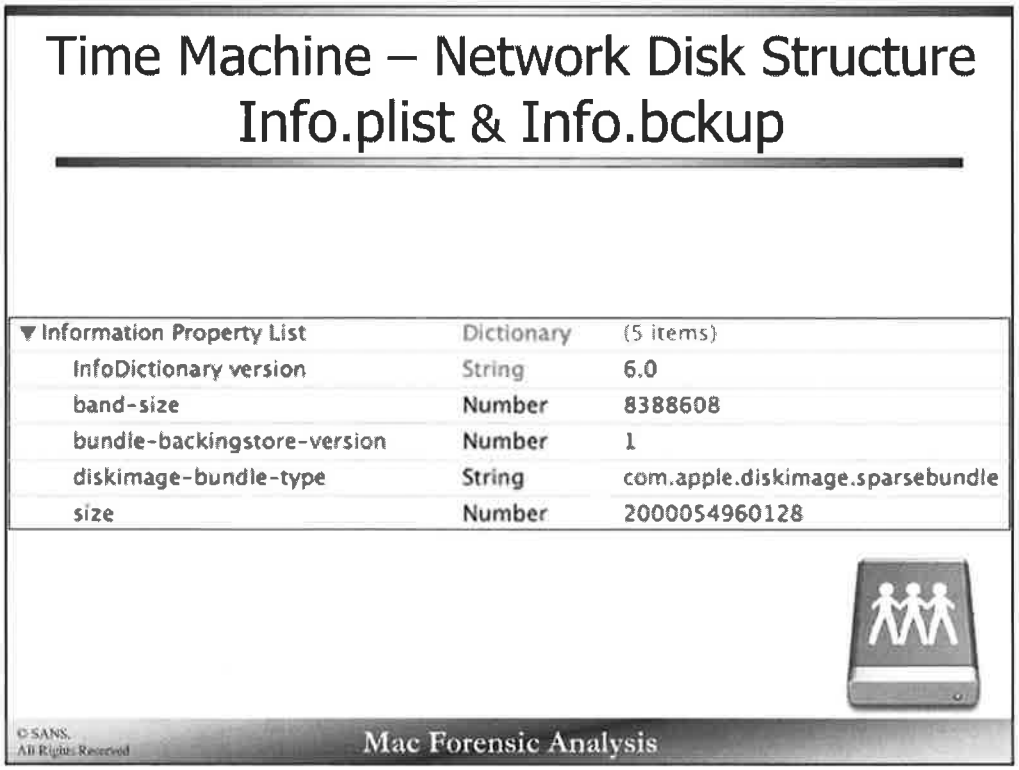

Data
FAVORITES
SHARED
delorean
hackintosh
readyshare
xbmc
DEVICES
Remote Disc
Data
Macintosh HD
Network
Remote Disc
bit.sparsebundle
byte.sparsebundle
nibble.blah.sparsebundle
Name bit.sparsebundle
Kind Sparse Disk Image Bundle
Size 1.35 TB
Created July 4, 2013 12:40 PM
Modified July 4, 2013 12:40 PM
Last opened July 4, 2013 12:40 PM



Each sparsebundle contains the following files and directories:

- `Info.plist` (and backup file)
- Bands directory
- `com.apple.TimeMachine.MachineID.plist` (and backup file)
- `com.apple.TimeMachine.Results.plist`
- `com.apple.TimeMachine.SnapshotHistory.plist`
- token file

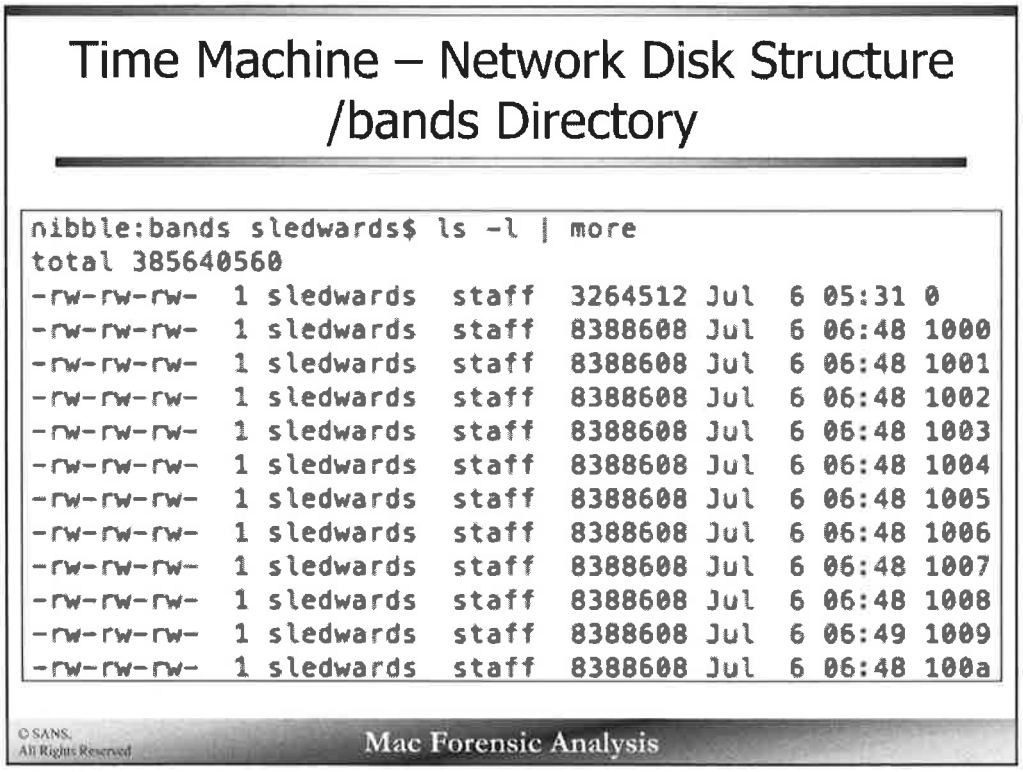
```
nibble:nibble.blah.sparsebundle sledwards$ ls -la
total 296
drwx------@    10 sledwards  staff     340 Jul  7 14:21 .
drwxrwxr-x     10 sledwards  staff     408 Jul  7 12:56 ..
-rw-r--r--      1 sledwards  staff     500 Jul  6 08:31 Info.bckup
-rw-r--r--      1 sledwards  staff     500 Jul  6 08:31 Info.plist
drwxr-xr-x  23544 sledwards  staff  800496 Jul  7 13:52 bands
-rw-r--r--      1 sledwards  staff     445 Jul  7 13:50 com.apple.TimeMachine.MachineID.bckup
-rw-r--r--      1 sledwards  staff     445 Jul  7 13:50 com.apple.TimeMachine.MachineID.plist
-rw-rw-rw-      1 sledwards  staff    1460 Jul  7 13:52 com.apple.TimeMachine.Results.plist
-rw-rw-rw-      1 sledwards  staff    4191 Jul  7 13:52 com.apple.TimeMachine.SnapshotHistory.plist
-rwx------      1 sledwards  staff  122368 Jul  6 08:31 token
```



`Info.plist` and `Info.bckup` are XML property lists and copies of each other. These property lists show the sparse bundle band size and allocated space of the sparse bundle . It is important to know the "allocated space" in the `size` key may be much larger than the actual disk that is backed up.

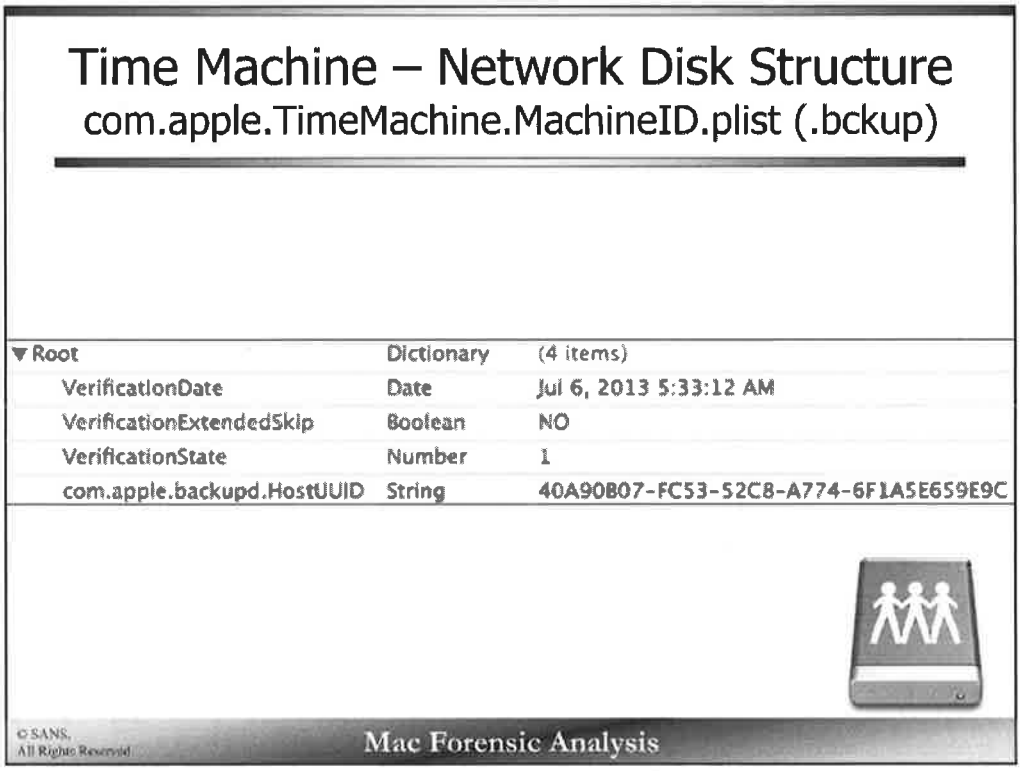
The band size is 8,388,608 bytes, or 8MB. The sparse bundle size is listed as 2,000,054,960,128 bytes or 2TB.

When this sparse bundle is mounted, it will show as a 2TB container. (The network backup volume used was 2TB in size.)

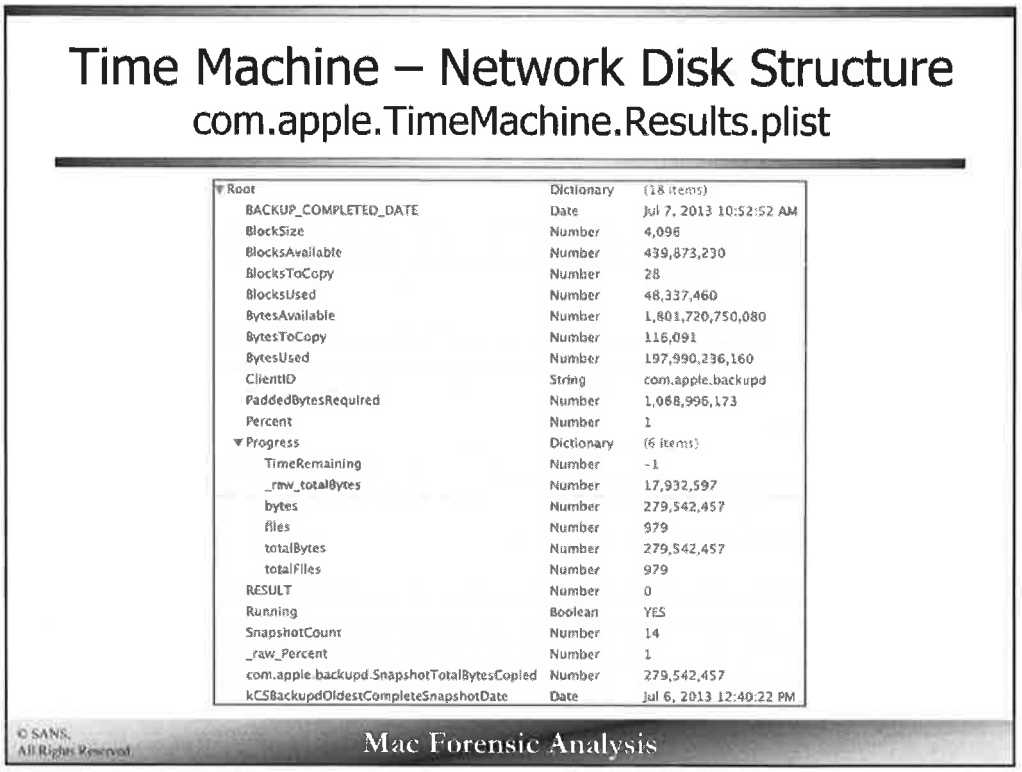


The bands directory contains the sparse bundle bands that create the encrypted volume. Each band has a maximum size of 8,388,608 bytes as indicated in the screenshot above. Each band file has a unique sequential alphanumeric name comprised of numbers 0-9 and the letters a-f.

The band files contain bits-and-pieces of the backed-up data in a non-human-readable format.



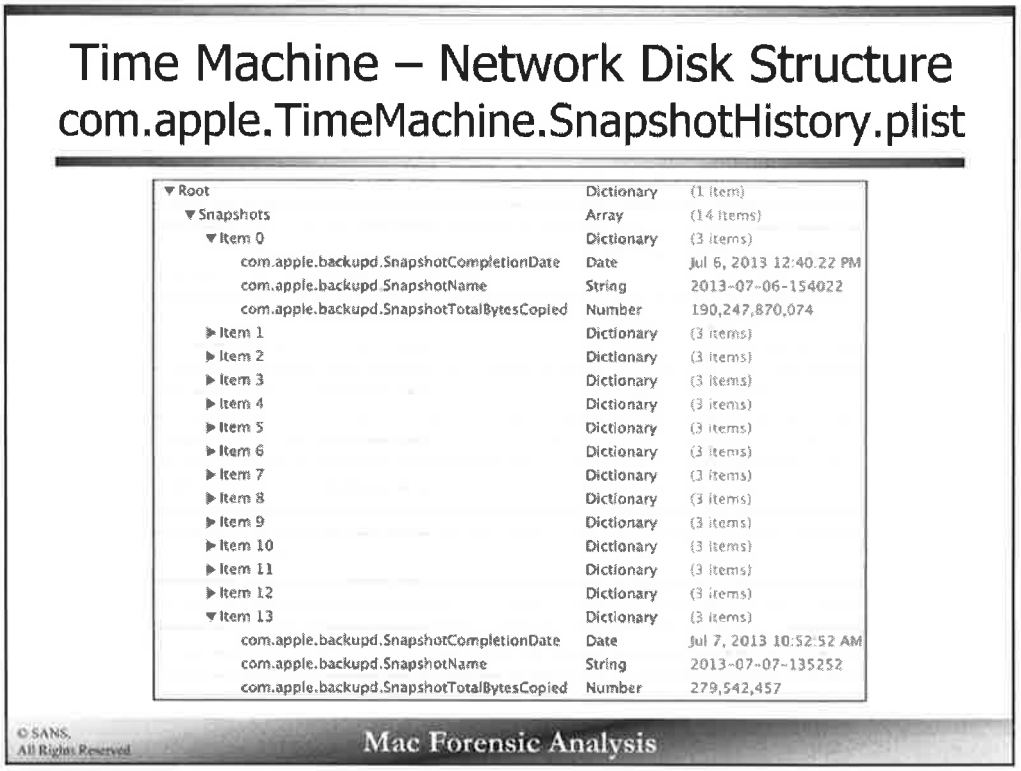
The property list files `com.apple.TimeMachine.MachineID.plist` and `com.apple.TimeMachine.MachineID.bckup` each contain the same information. These files contain the Host UUID, and Time Machine verification data.



The `com.apple.TimeMachine.Results.plist` contains data about the backup process including:

- Backup Completion Date
- Disk block size and availability
- Disk byte size and availability
- Backup Progress
- Snapshot Data

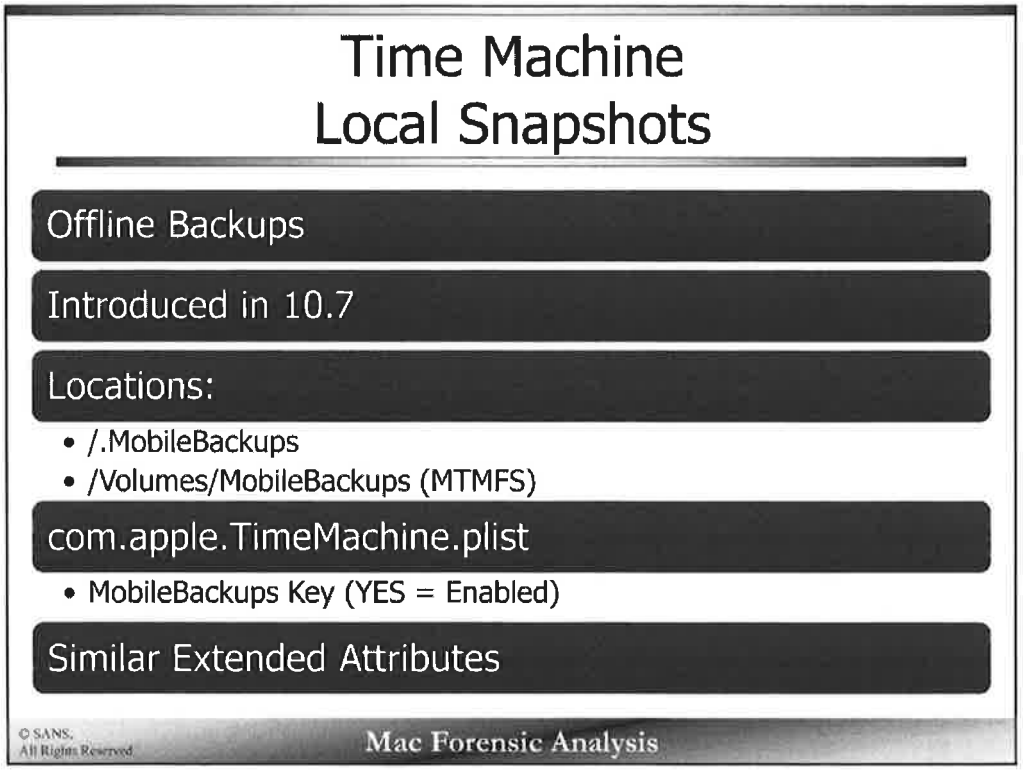
▼ Root	Dictionary	(18 items)
BACKUP_COMPLETED_DATE	Date	Jul 7, 2013 10:52:52 AM
BlockSize	Number	4,096
BlocksAvailable	Number	439,873,230
BlocksToCopy	Number	28
BlocksUsed	Number	48,337,460
BytesAvailable	Number	1,801,720,750,080
BytesToCopy	Number	116,091
BytesUsed	Number	197,990,236,160
ClientID	String	com.apple.backupd
PaddedBytesRequired	Number	1,068,996,173
Percent	Number	1
▼ Progress	Dictionary	(6 items)
TimeRemaining	Number	-1
_raw_totalBytes	Number	17,932,597
bytes	Number	279,542,457
files	Number	979
totalBytes	Number	279,542,457
totalFiles	Number	979
RESULT	Number	0
Running	Boolean	YES
SnapshotCount	Number	14
_raw_Percent	Number	1
com.apple.backupd.SnapshotTotalBytesCopied	Number	279,542,457
kCSBackupdOldestCompleteSnapshotDate	Date	Jul 6, 2013 12:40:22 PM



The `com.apple.TimeMachine.SnapshotHistory.plist` property list contains data about the stored Time Machine snapshots. This backup contains 14 snapshots as shown in the screenshot above. Each snapshot `Item` has three keys associated:

- `com.apple.backupd.SnapshotCompletionDate` – Snapshot completion timestamp
- `com.apple.backupd.SnapshotName` – Snapshot Name when sparse bundle is mounted
- `com.apple.backupd.SnapshotTotalBytesCopied` – Bytes copied by snapshot

▼ Root	Dictionary	(1 item)
▼ Snapshots	Array	(14 items)
▼ Item 0	Dictionary	(3 items)
com.apple.backupd.SnapshotCompletionDate	Date	Jul 6, 2013 12:40:22 PM
com.apple.backupd.SnapshotName	String	2013-07-06-154022
com.apple.backupd.SnapshotTotalBytesCopied	Number	190,247,870,074
▶ Item 1	Dictionary	(3 items)
▶ Item 2	Dictionary	(3 items)
▶ Item 3	Dictionary	(3 items)
▶ Item 4	Dictionary	(3 items)
▶ Item 5	Dictionary	(3 items)
▶ Item 6	Dictionary	(3 items)
▶ Item 7	Dictionary	(3 items)
▶ Item 8	Dictionary	(3 items)
▶ Item 9	Dictionary	(3 items)
▶ Item 10	Dictionary	(3 items)
▶ Item 11	Dictionary	(3 items)
▶ Item 12	Dictionary	(3 items)
▼ Item 13	Dictionary	(3 items)
com.apple.backupd.SnapshotCompletionDate	Date	Jul 7, 2013 10:52:52 AM
com.apple.backupd.SnapshotName	String	2013-07-07-135252
com.apple.backupd.SnapshotTotalBytesCopied	Number	279,542,457



When the Time Machine volume is otherwise inaccessible due to lack of network access or the time machine volume is not mounted, the system (if configured) will create local snapshots of the system. These are similar in format to normal Time Machine backups; however, they are saved to the local hard drive (if space permits).

This functionality was introduced in 10.7. This feature may be enabled on laptops and is disabled on desktop systems by default.

You may find these artifacts in two directories depending if you are looking at a forensic image or a live system.

- `/.MobileBackups` – This directory will exist in the root of the system, for example in a forensic image of a system.
- `/Volumes/MobileBackups` – On a live system the `MobileBackups` volume may be mounted. This is a Mobile Time Machine File System (MTMFS) volume.

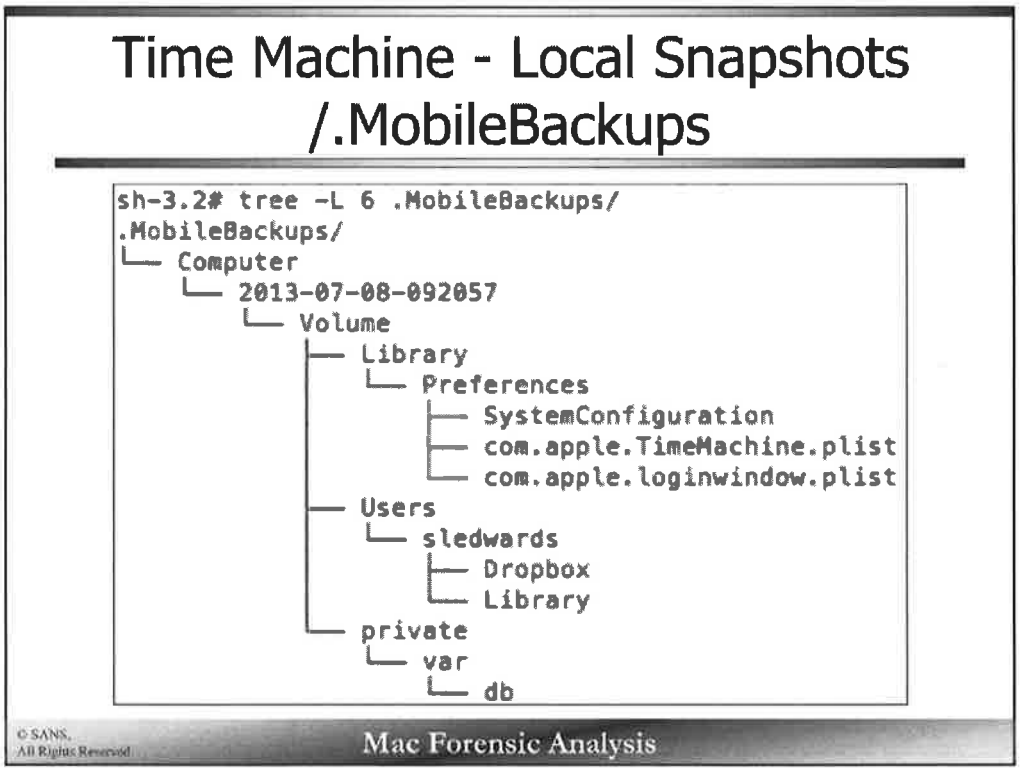
References:

http://pondini.org/TM/30.html

mtmd Man Page

mtmfs Man Page

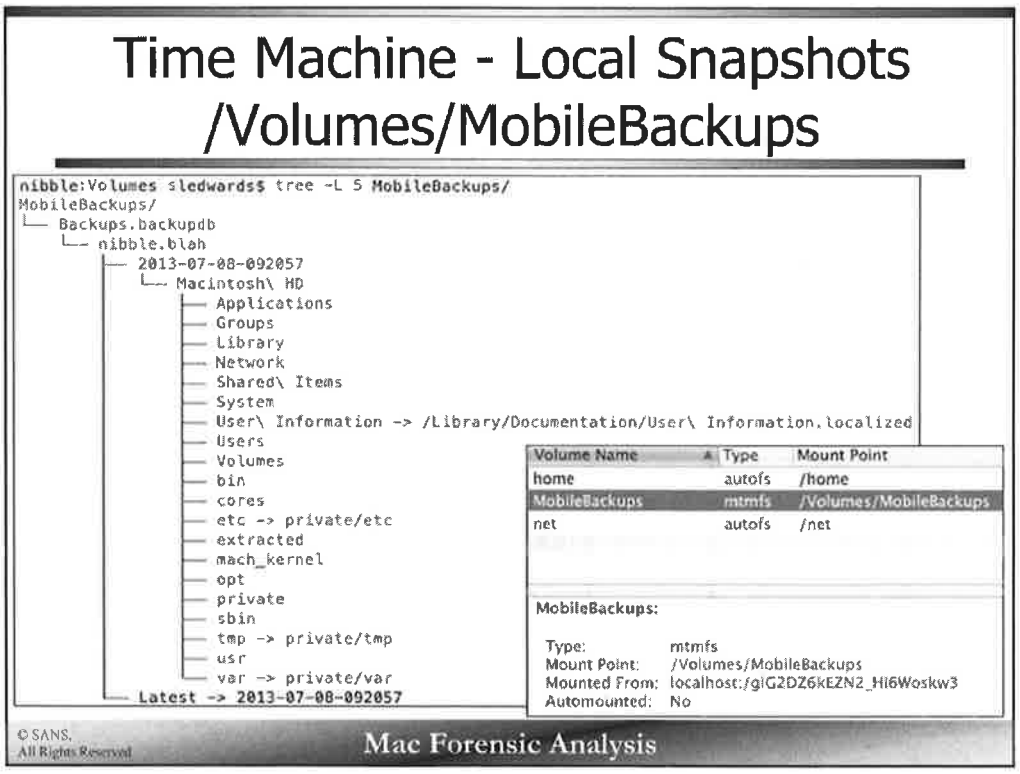
http://support.apple.com/kb/HT4878



The `/.MobileBackups` directory in the root of the system volume contains a slightly different format than the `MobileBackups` volume. It will only contain those files that have been changed with no hard links to the rest of the file system. Instead of '`nibble.blah`' as in previous examples, it uses the more generic '`Computer`'.

References:

http://pondini.org/TM/30.html



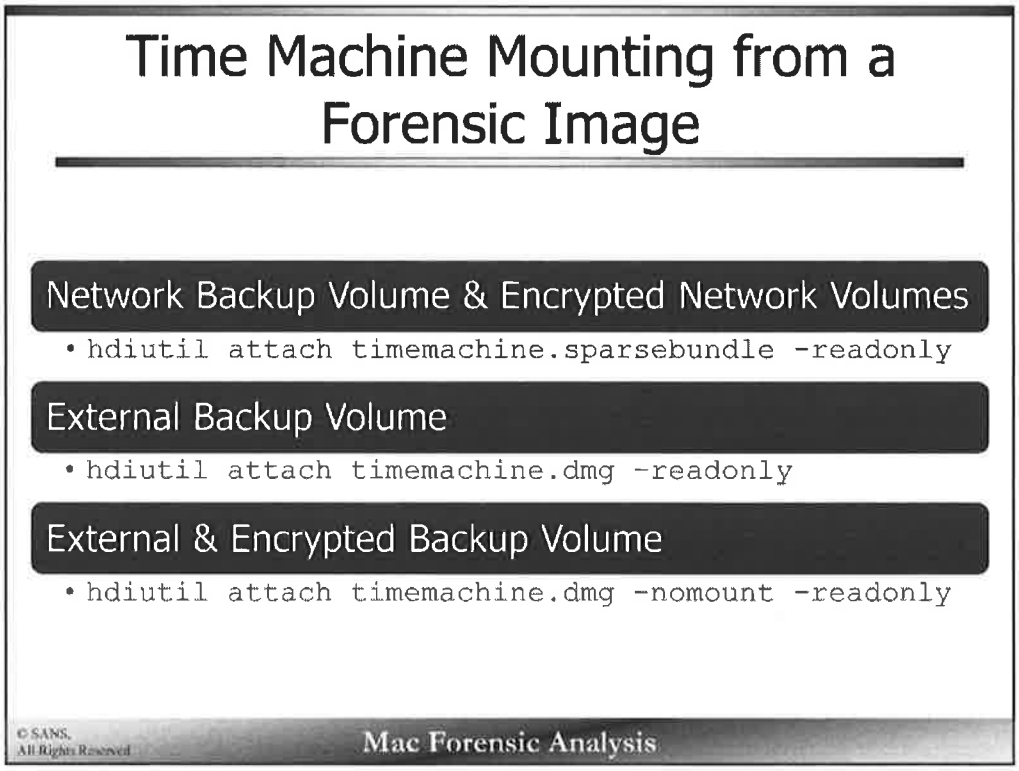
The mounted `MobileBackups` volume is a '`mtmfs`' volume meaning it is a `Mobile Time Machine File System`.

The `tree` output in the screenshot above shows the backup uses the same format as the normal Time Machine backups. It includes hard links to full file system.

The `tree` command is not installed by default, but can be downloaded from http://mama.indstate.edu/users/ice/tree/ or installed via MacPorts, HomeBrew, or Fink.

References:

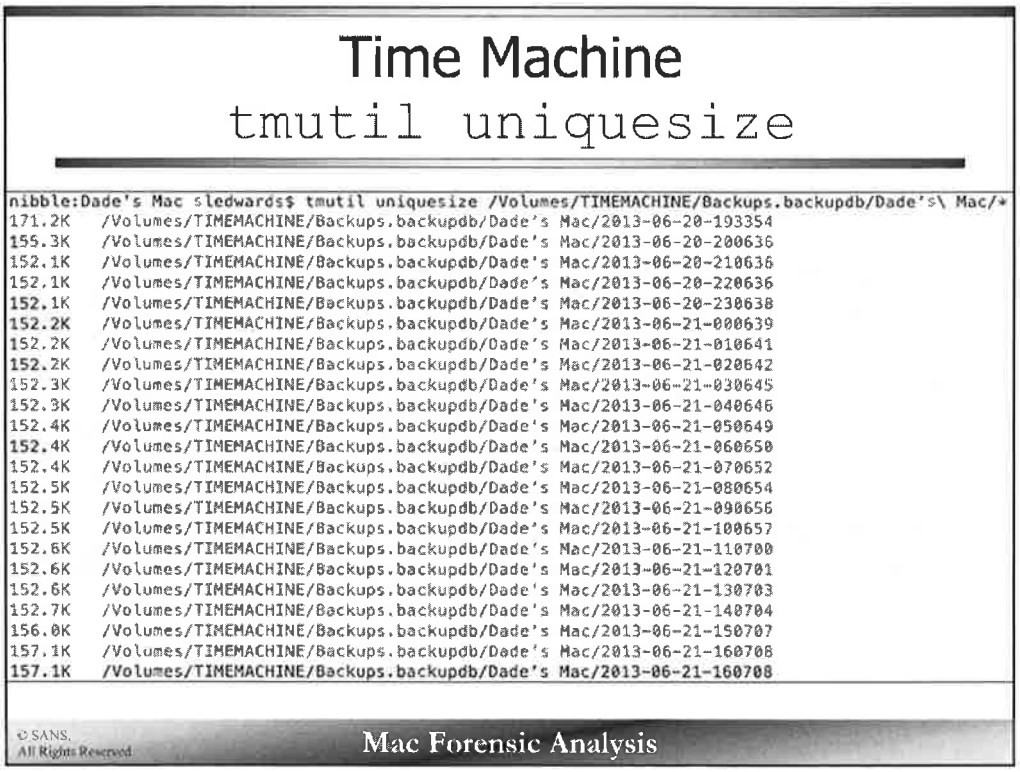
http://pondini.org/TM/30.html



We can use the `hdiutil attach` command to mount these Time Machine images using the `-readonly` flag to be forensically sound.

This allows us to view the files inside the images as the system would have seen them. We can view all the files, hard links, and various snapshots. We can then use native OS X tools to start analyzing these files.

Each command uses nearly the same format, but the parameters have been adapted for the particular image type.



The `tmutil` command line utility interacts with Time Machine backups and may have to run with root privileges. We can use this utility to gather information about the backups and compare snapshots.

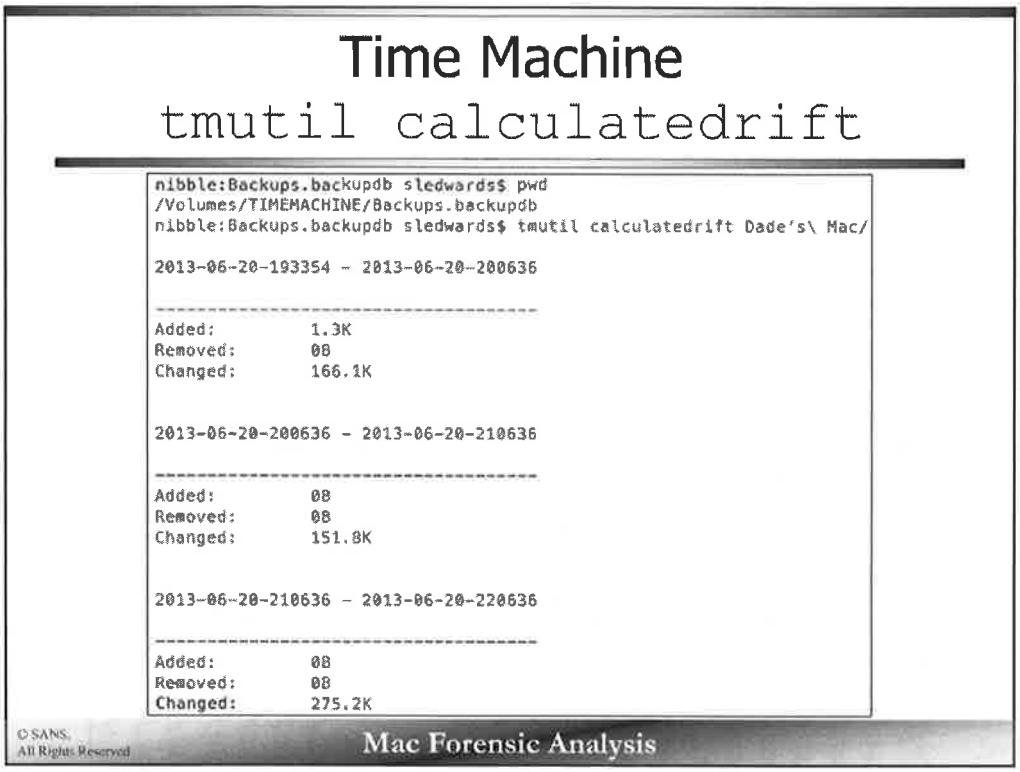
This utility can also be used on a system to change time machine preferences, start backups, restore files, or to delete backups.

The verb `uniquesize` used with the `tmutil` utility will show the unique sizes of each snapshot, minus the hard-linked data. This is the true size of each snapshot backup.

References:

`tmutil` Man Page

```
nibble:Dade's Mac sledwards$ tmutil uniquesize /Volumes/TIMEMACHINE/Backups.backupdb/Dade's\ Mac/*
171.2K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-193354
155.3K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636
152.1K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-210636
152.1K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-220636
152.1K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-230638
152.2K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-000639
152.2K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-010641
152.2K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-020642
152.3K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-030645
152.3K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-040646
152.4K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-050649
152.4K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-060650
152.4K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-070652
152.5K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-080654
152.5K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-090656
152.5K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-100657
152.6K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-110700
152.6K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-120701
152.6K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-130703
152.7K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-140704
156.0K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-150707
157.1K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-160708
157.1K   /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-21-160708
```



The verb `calculatedrift` used with the `tmutil` utility will show the size of changes between each snapshot. Each calculation includes the amount of data added, removed, and/or changed.

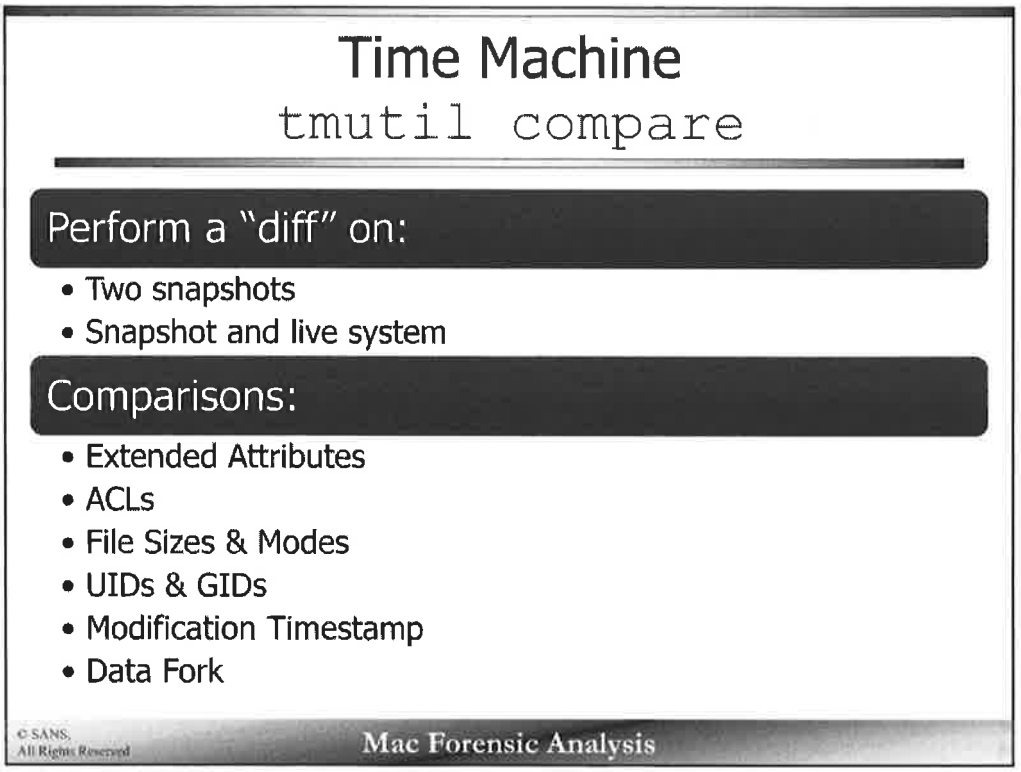
For example, between the snapshots `2013-06-20-193354` and `2013-06-20-200636`, 1.3K was added, nothing was removed, and 166.1K was changed. The "`tmutil calculatedrift`" command is used on the machine directory, in this example "`Dade's Mac`". This command will show the changes for each snapshot in this directory. At the completion of the command it will show "`Drift Averages`", or averages of each addition, removal, or change.

This command is useful to determine what snapshot may have the most data additions, deletions, and changes on a system. Perhaps a large amount of data was removed from the system. This command is a quick test to determine which two snapshots to examine first to see what data was removed.

References:

`tmutil` Man Page

```
nibble:Backups.backupdb sledwards$ pwd
/Volumes/TIMEMACHINE/Backups.backupdb
nibble:Backups.backupdb sledwards$ tmutil calculatedrift Dade's\ Mac/

2013-06-20-193354 - 2013-06-20-200636

----------------------------------------
Added:          1.3K
Removed:        0B
Changed:        166.1K


2013-06-20-200636 - 2013-06-20-210636

----------------------------------------
Added:          0B
Removed:        0B
Changed:        151.8K


2013-06-20-210636 - 2013-06-20-220636

----------------------------------------
Added:          0B
Removed:        0B
Changed:        275.2K
```



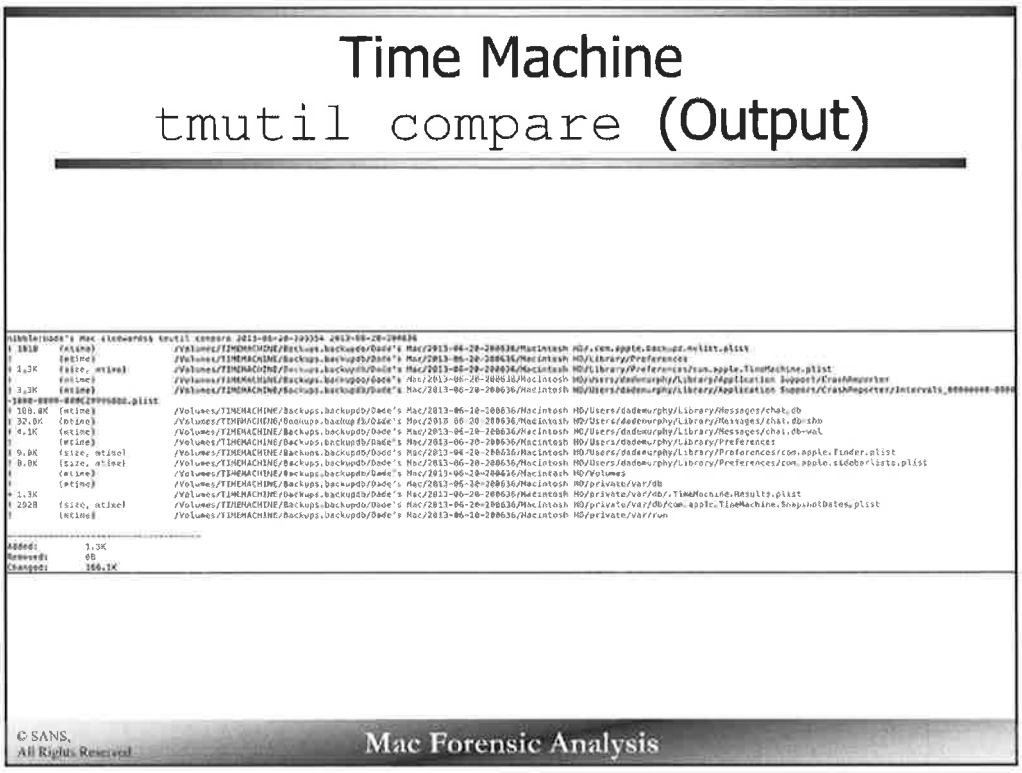
The verb `compare` used with the `tmutil` utility will perform a diff between snapshots or a snapshot and the live system.

By default it will compare the following values:

- File Size
- File Mode
- UID
- GID
- Modification Time

References:

`tmutil` Man Page



The default output for the `tmutil compare` command is shown in the screenshot above. Each change is labeled with one of the following marks in the first column:

- ! Metadata Change
- + File Creation
- - File Removal

The second column shows the file size of the newest file.

The third column contains what metadata was changed.

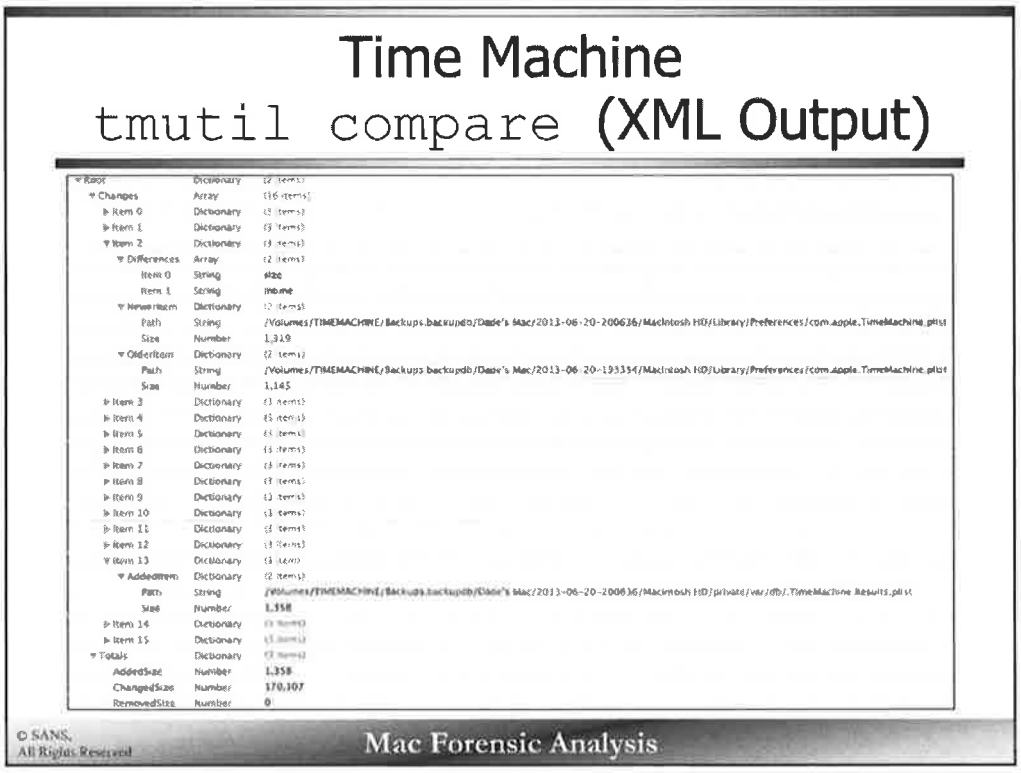
The fourth column contains the file path to the specific file.

The last output contains the overall changes between the snapshots.

References:

`tmutil` Man Page

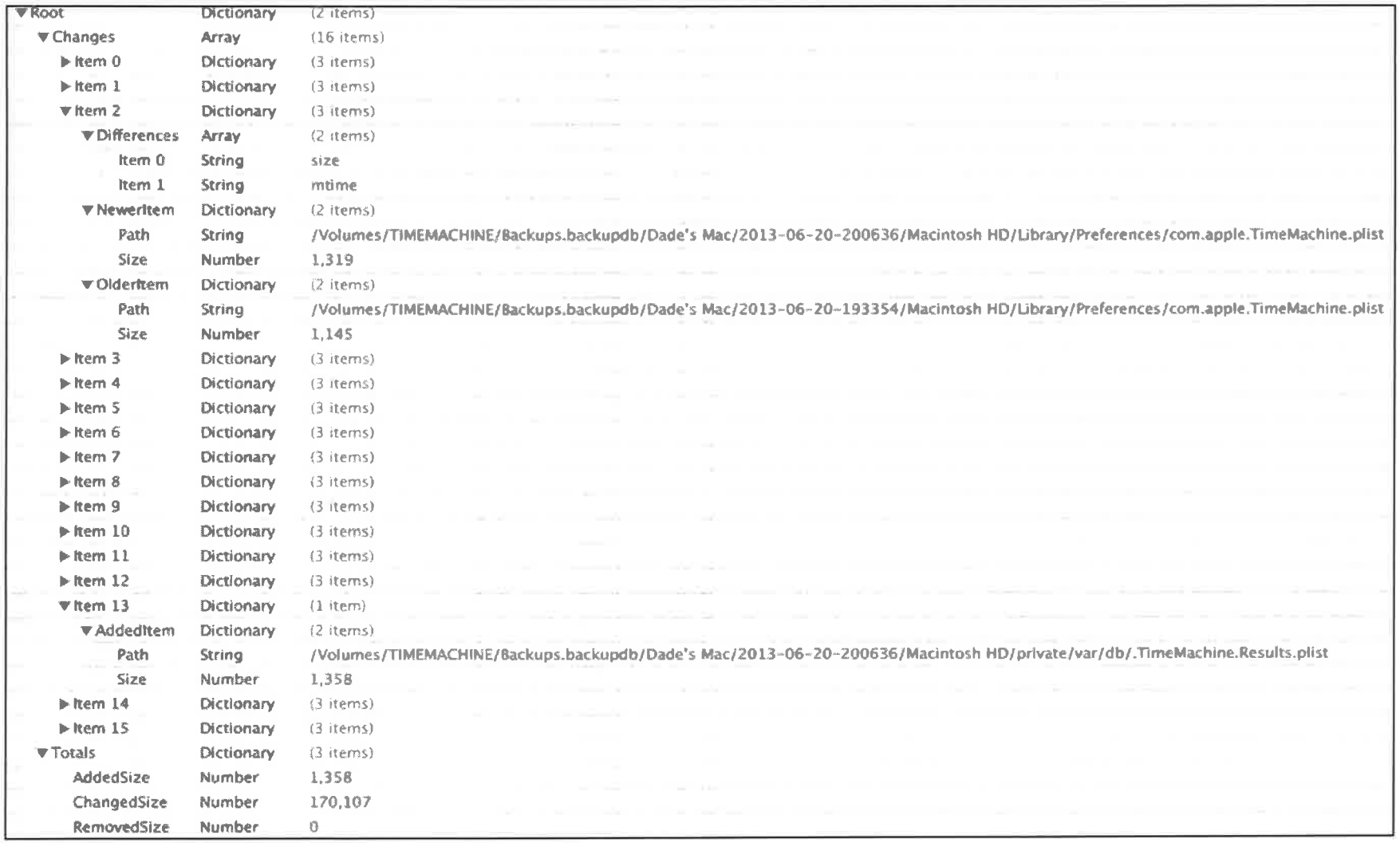
```
nibble:Dade's Mac sledwards$ tmutil compare 2013-06-20-193354 2013-06-20-200636
! 1818    (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/.com.apple.backupd.mvlist.plist
!         (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Library/Preferences
! 1.3K    (size, mtime)       /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Library/Preferences/com.apple.TimeMachine.plist
!         (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Users/dademurphy/Library/Application Support/CrashReporter
! 3.3K    (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Users/dademurphy/Library/Application Support/CrashReporter/Intervals_00000000-0000
-1000-8000-000C2999688E.plist
! 108.0K  (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Users/dademurphy/Library/Messages/chat.db
! 32.0K   (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Users/dademurphy/Library/Messages/chat.db-shm
! 4.1K    (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Users/dademurphy/Library/Messages/chat.db-wal
!         (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Users/dademurphy/Library/Preferences
! 9.0K    (size, mtime)       /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Users/dademurphy/Library/Preferences/com.apple.finder.plist
! 8.0K    (size, mtime)       /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Users/dademurphy/Library/Preferences/com.apple.sidebarlists.plist
!         (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Volumes
!         (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/private/var/db
+ 1.3K                       /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/private/var/db/.TimeMachine.Results.plist
! 2928    (size, mtime)       /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/private/var/db/com.apple.TimeMachine.SnapshotDates.plist
!         (mtime)             /Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/private/var/run

-----------------------------------
Added:        1.3K
Removed:      0B
Changed:      166.1K
```



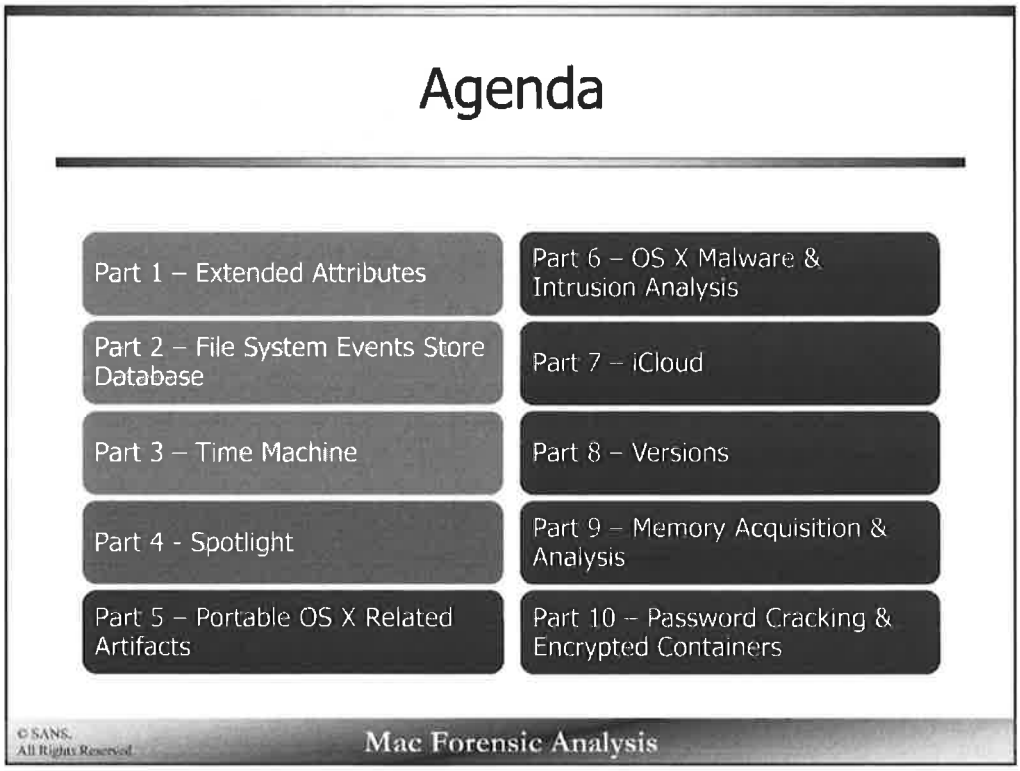
The `tmutil compare` command can also be output to an XML file for an easier and slightly more detailed view. This format can be viewed in Xcode, and file size changes are shown from the older file to the newer file.

References:

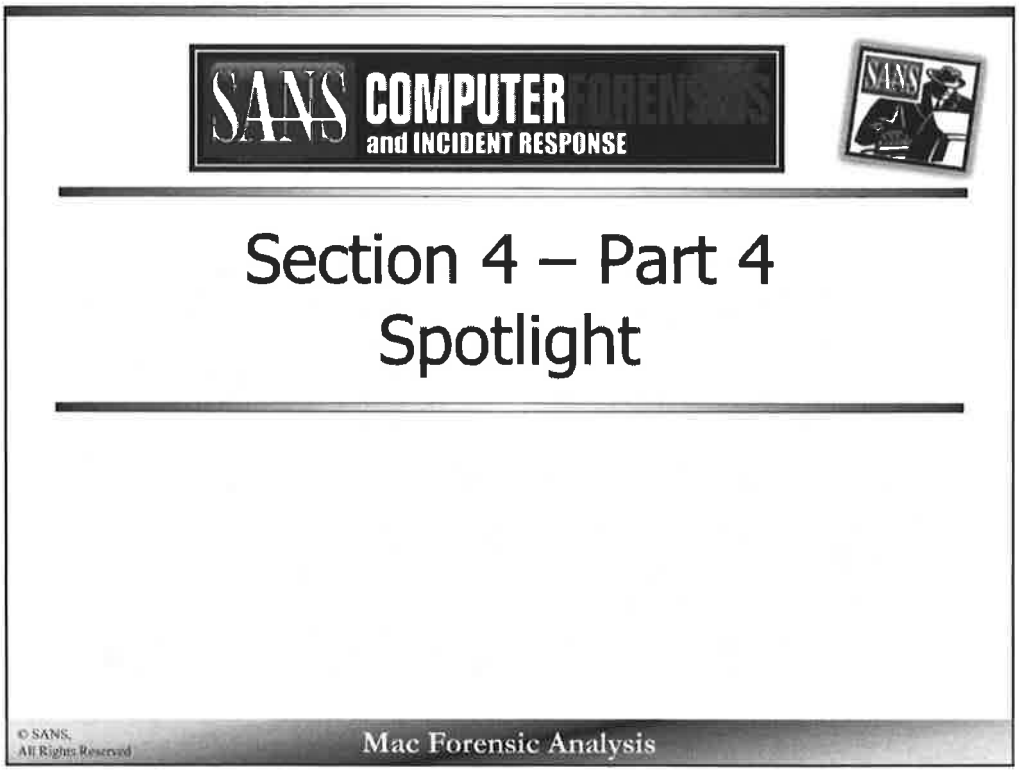
`tmutil` Man Page



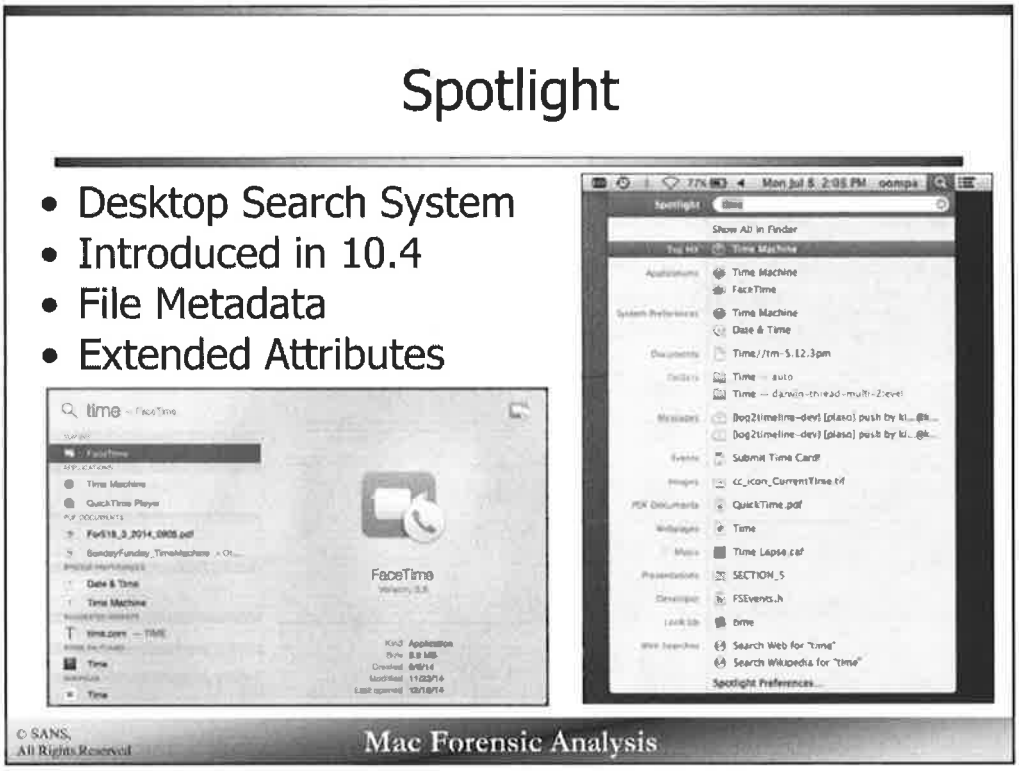
▼Root	Dictionary	(2 items)
▼Changes	Array	(16 items)
▶Item 0	Dictionary	(3 items)
▶Item 1	Dictionary	(3 items)
▼Item 2	Dictionary	(3 items)
▼Differences	Array	(2 items)
Item 0	String	size
Item 1	String	mtime
▼NewerItem	Dictionary	(2 items)
Path	String	/Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/Library/Preferences/com.apple.TimeMachine.plist
Size	Number	1,319
▼OlderItem	Dictionary	(2 items)
Path	String	/Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-193354/Macintosh HD/Library/Preferences/com.apple.TimeMachine.plist
Size	Number	1,145
▶Item 3	Dictionary	(3 items)
▶Item 4	Dictionary	(3 items)
▶Item 5	Dictionary	(3 items)
▶Item 6	Dictionary	(3 items)
▶Item 7	Dictionary	(3 items)
▶Item 8	Dictionary	(3 items)
▶Item 9	Dictionary	(3 items)
▶Item 10	Dictionary	(3 items)
▶Item 11	Dictionary	(3 items)
▶Item 12	Dictionary	(3 items)
▼Item 13	Dictionary	(1 item)
▼AddedItem	Dictionary	(2 items)
Path	String	/Volumes/TIMEMACHINE/Backups.backupdb/Dade's Mac/2013-06-20-200636/Macintosh HD/private/var/db/.TimeMachine.Results.plist
Size	Number	1,358
▶Item 14	Dictionary	(3 items)
▶Item 15	Dictionary	(3 items)
▼Totals	Dictionary	(3 items)
AddedSize	Number	1,358
ChangedSize	Number	170,107
RemovedSize	Number	0



This page intentionally left blank.



This page intentionally left blank.



OS X uses Spotlight to index the system to allow the user to search for applications, documents, and other files quickly. This feature was introduced in OS X 10.4 and on iPhones with iOS 3. File metadata, including the extended attributes are all indexed.

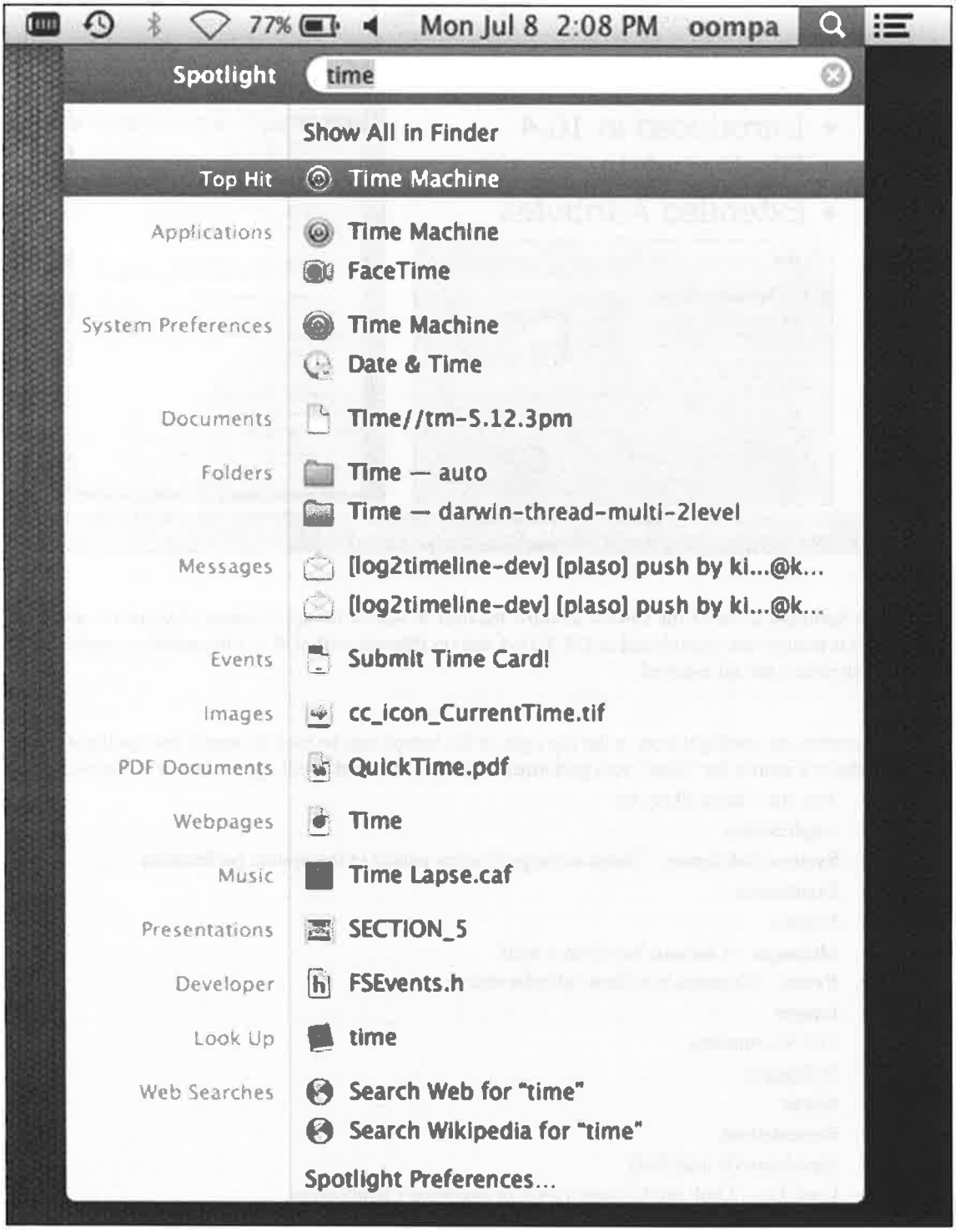
On a live system the spotlight icon in the top right of the screen can be used to search the Spotlight index. In the example above a search for "time" was performed. Live search results will appear below in various categories:

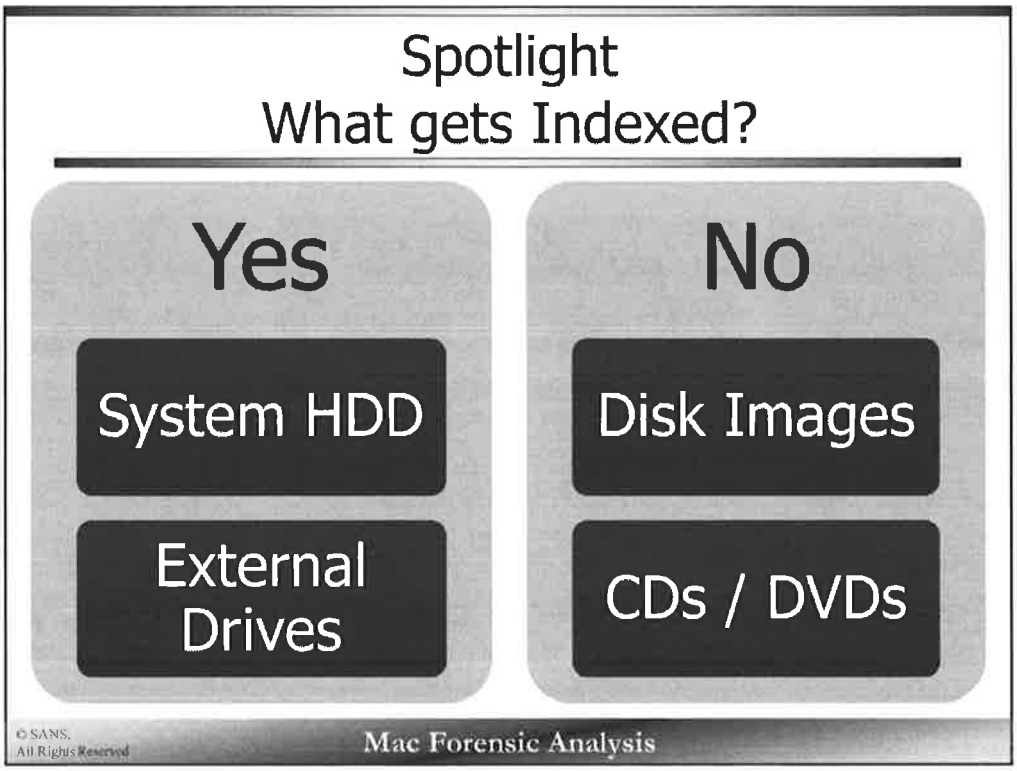
- Top Hit – Most likely hit
- Applications
- System Preferences – Items in the preference panels in the system preferences
- Documents
- Folders
- Messages – Contains hits from e-mail
- Events – Contains hits from calendar entries
- Images
- PDF Documents
- Webpages
- Music
- Presentations
- Developer (if installed)
- Look Up – Look up the search term in Dictionary application
- Web Searches – Perform a web search for the search term

References:

Mac OS X and iOS Internals: To the Apple's Core

Chapter 2 – The User Experience Layer


77%
Mon Jul 8 2:08 PM
oompa
Spotlight
time
Show All in Finder
Top Hit
Time Machine
Applications
Time Machine
FaceTime
System Preferences
Time Machine
Date & Time
Documents
Time//tm-5.12.3pm
Folders
Time — auto
Time — darwin-thread-multi-2level
Messages
[log2timeline-dev] [plaso] push by ki...@k...
[log2timeline-dev] [plaso] push by ki...@k...
Events
Submit Time Card!
Images
cc_icon_CurrentTime.tif
PDF Documents
QuickTime.pdf
Webpages
Time
Music
Time Lapse.caf
Presentations
SECTION_5
Developer
FSEvents.h
Look Up
time
Web Searches
Search Web for "time"
Search Wikipedia for "time"
Spotlight Preferences...



While system hard drives and external drives are indexed, other locations may not be indexed by default. These locations include DMG files, CDs and DVDs, hidden files and system directories.

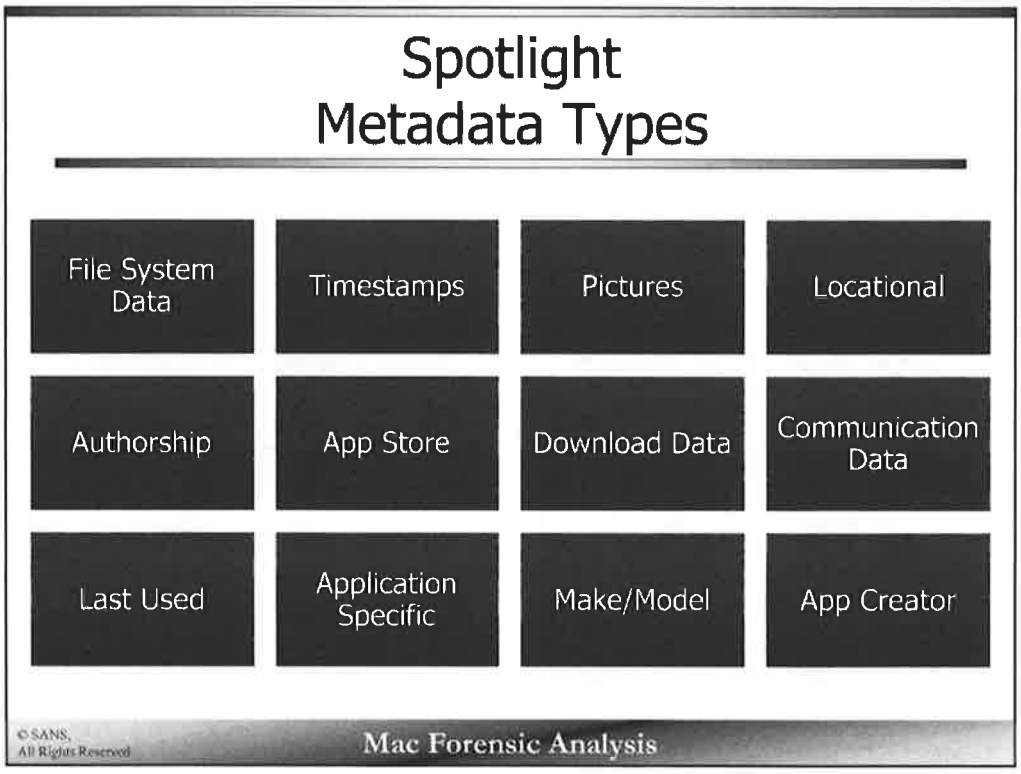
A volume can explicitly be told not to be indexed by placing an empty, hidden file with the filename `.metadata_never_index` at the root of the volume.

A volume can be set not to index by using the `mdutil` command.

References:

http://www.thexlab.com/faqs/stopspotlightindex.html

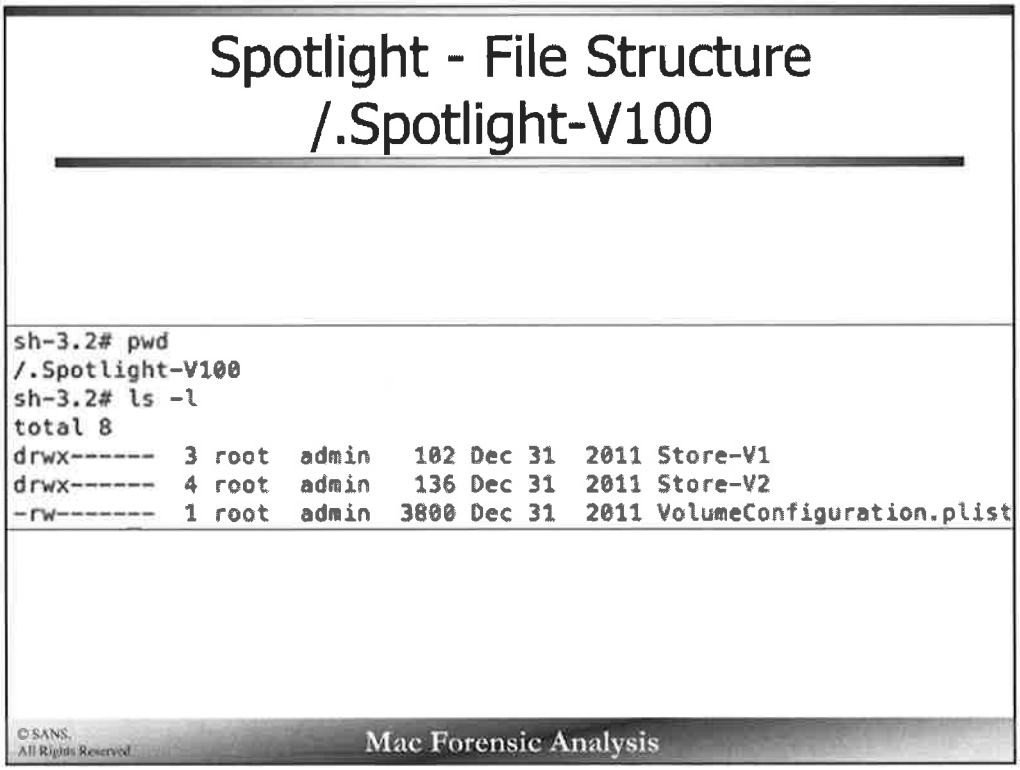
`mdutil` Man Page



What type of data gets indexed? Anything and everything!

An small example of what might be indexed:

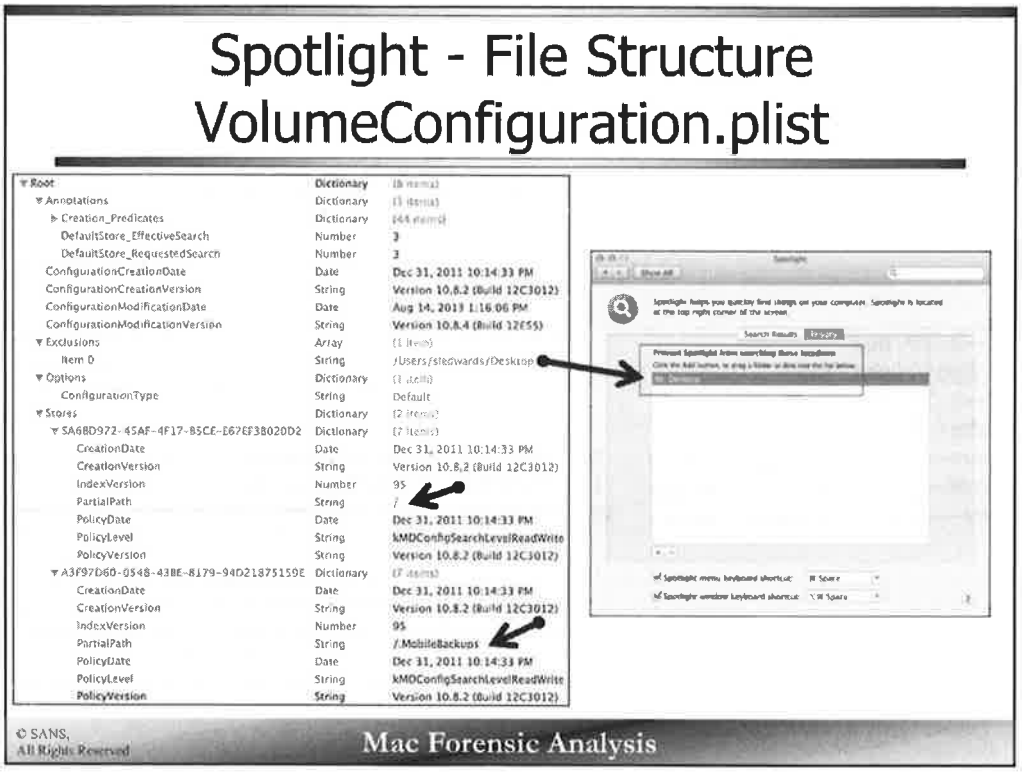
- Another copy of file system metadata, including file names, logical and physical file sizes, UID/GID, and file system timestamps.
- Timestamps – Along with the file system timestamps, you may also find download dates, last used dates, and date added dates.
- Photos data may include, width/length, make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



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

`Store-V1` is used with older OS X versions (10.6 and below), while the second version of the store `Store-V2` was introduced in 10.7.

You will see both directories on newer versions of OS X as shown in the screenshot above.



The `.Spotlight-V100` directory contains the property list `VolumeConfiguration.plist`. This property list contains indexing exclusions (amongst other Spotlight configuration data). In the example above, the user's desktop was excluded from indexing.

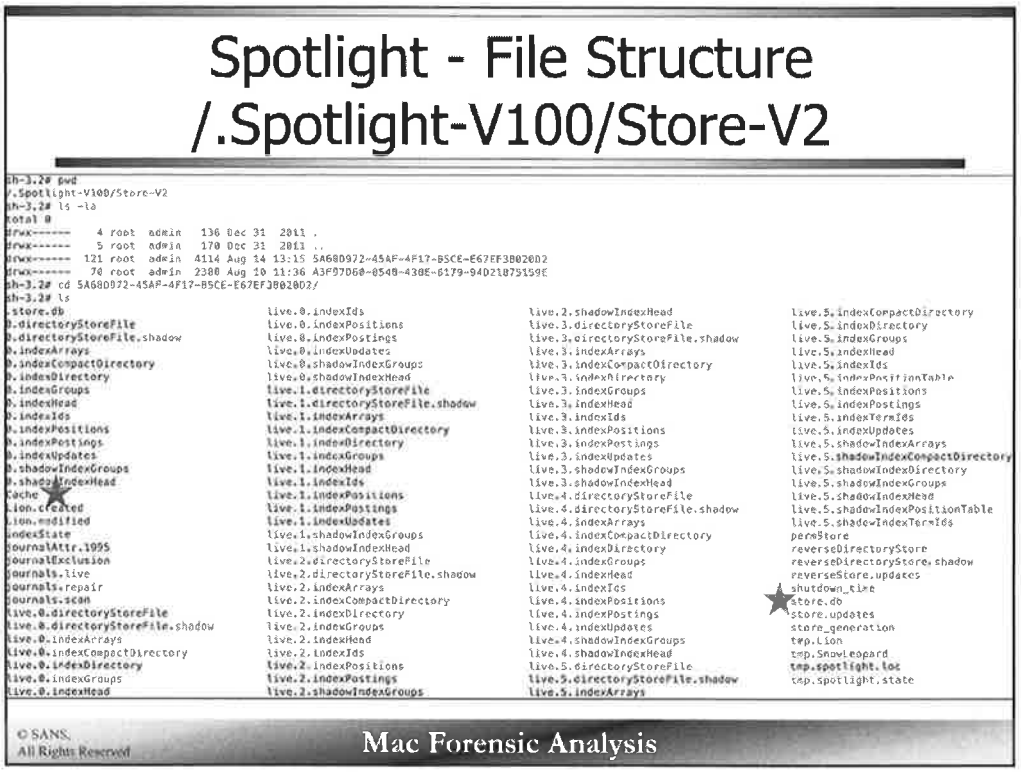
The `ConfigurationModificationDate` key contains the timestamp when the Spotlight configuration was last modified.

The two GUIDs in the example, show that two volumes are currently being indexed:

- `/` - The root volume, and
- `/.MobileBackups` – a Time Machine related volume.

The GUIDs reference the Spotlight Stores where the volume indexing database is stored.

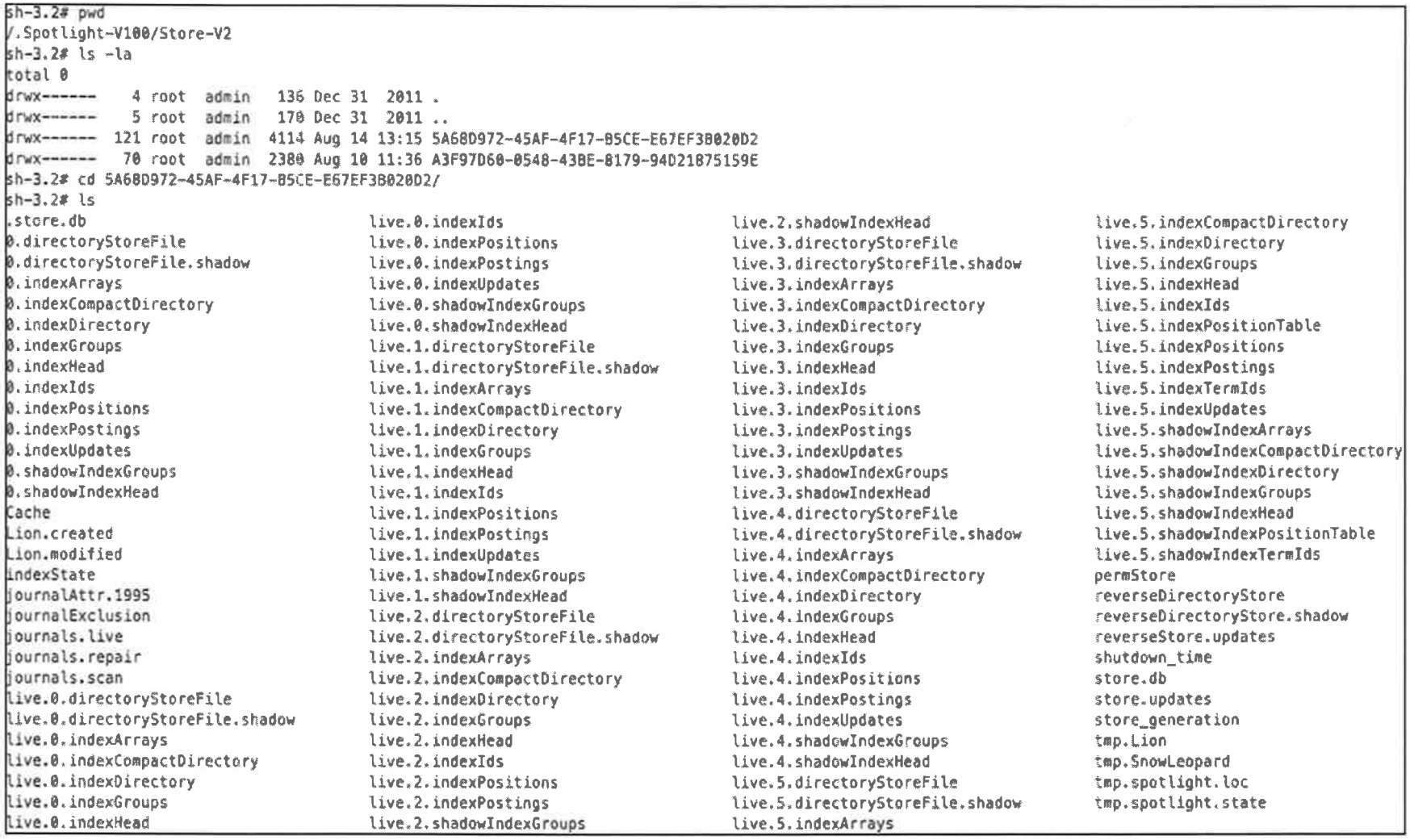
▼ Root	Dictionary	(8 items)
▼ Annotations	Dictionary	(3 items)
▶ Creation_Predicates	Dictionary	(44 items)
DefaultStore_EffectiveSearch	Number	3
DefaultStore_RequestedSearch	Number	3
ConfigurationCreationDate	Date	Dec 31, 2011 10:14:33 PM
ConfigurationCreationVersion	String	Version 10.8.2 (Build 12C3012)
ConfigurationModificationDate	Date	Aug 14, 2013 1:16:06 PM
ConfigurationModificationVersion	String	Version 10.8.4 (Build 12E55)
▼ Exclusions	Array	(1 item)
Item 0	String	/Users/sledwards/Desktop
▼ Options	Dictionary	(1 item)
ConfigurationType	String	Default
▼ Stores	Dictionary	(2 items)
▼ 5A68D972-45AF-4F17-B5CE-E67EF3B020D2	Dictionary	(7 items)
CreationDate	Date	Dec 31, 2011 10:14:33 PM
CreationVersion	String	Version 10.8.2 (Build 12C3012)
IndexVersion	Number	95
PartialPath	String	/
PolicyDate	Date	Dec 31, 2011 10:14:33 PM
PolicyLevel	String	kMDConfigSearchLevelReadWrite
PolicyVersion	String	Version 10.8.2 (Build 12C3012)
▼ A3F97D60-0548-43BE-8179-94D21875159E	Dictionary	(7 items)
CreationDate	Date	Dec 31, 2011 10:14:33 PM
CreationVersion	String	Version 10.8.2 (Build 12C3012)
IndexVersion	Number	95
PartialPath	String	/.MobileBackups
PolicyDate	Date	Dec 31, 2011 10:14:33 PM
PolicyLevel	String	kMDConfigSearchLevelReadWrite
PolicyVersion	String	Version 10.8.2 (Build 12C3012)



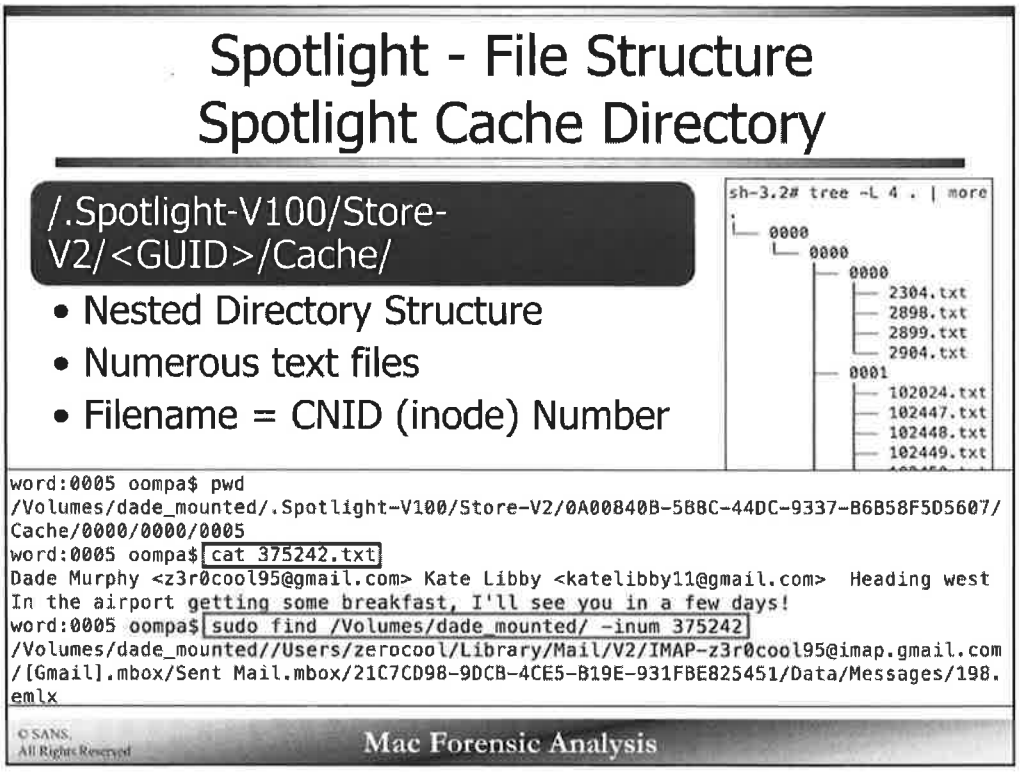
Each Spotlight Store directory contains many files.

The example above is from a Mountain Lion OS X system, these files are the indexing databases used by Spotlight. The format for most of these files is unknown at this time.

The two places were data is readily available for investigators is the `Cache` directory and the `store.db` index database.



```
sh-3.2# pwd
/.Spotlight-V100/Store-V2
sh-3.2# ls -la
total 0
drwx------    4 root  admin   136 Dec 31  2011 .
drwx------    5 root  admin   170 Dec 31  2011 ..
drwx------  121 root  admin  4114 Aug 14 13:15 5A68D972-45AF-4F17-B5CE-E67EF3B020D2
drwx------   70 root  admin  2380 Aug 10 11:36 A3F97D60-0548-43BE-8179-94D21875159E
sh-3.2# cd 5A68D972-45AF-4F17-B5CE-E67EF3B020D2/
sh-3.2# ls
.store.db                           live.0.indexIds                     live.2.shadowIndexHead              live.5.indexCompactDirectory
0.directoryStoreFile                live.0.indexPositions               live.3.directoryStoreFile           live.5.indexDirectory
0.directoryStoreFile.shadow         live.0.indexPostings                live.3.directoryStoreFile.shadow    live.5.indexGroups
0.indexArrays                       live.0.indexUpdates                 live.3.indexArrays                  live.5.indexHead
0.indexCompactDirectory             live.0.shadowIndexGroups            live.3.indexCompactDirectory        live.5.indexIds
0.indexDirectory                    live.0.shadowIndexHead              live.3.indexDirectory               live.5.indexPositionTable
0.indexGroups                       live.1.directoryStoreFile           live.3.indexGroups                  live.5.indexPositions
0.indexHead                         live.1.directoryStoreFile.shadow    live.3.indexHead                    live.5.indexPostings
0.indexIds                          live.1.indexArrays                  live.3.indexIds                     live.5.indexTermIds
0.indexPositions                    live.1.indexCompactDirectory        live.3.indexPositions               live.5.indexUpdates
0.indexPostings                     live.1.indexDirectory               live.3.indexPostings                live.5.shadowIndexArrays
0.indexUpdates                      live.1.indexGroups                  live.3.indexUpdates                 live.5.shadowIndexCompactDirectory
0.shadowIndexGroups                 live.1.indexHead                    live.3.shadowIndexGroups            live.5.shadowIndexDirectory
0.shadowIndexHead                   live.1.indexIds                     live.3.shadowIndexHead              live.5.shadowIndexGroups
Cache                               live.1.indexPositions               live.4.directoryStoreFile           live.5.shadowIndexHead
Lion.created                        live.1.indexPostings                live.4.directoryStoreFile.shadow    live.5.shadowIndexPositionTable
Lion.modified                       live.1.indexUpdates                 live.4.indexArrays                  live.5.shadowIndexTermIds
indexState                          live.1.shadowIndexGroups            live.4.indexCompactDirectory        permStore
journalAttr.1995                    live.1.shadowIndexHead              live.4.indexDirectory               reverseDirectoryStore
journalExclusion                    live.2.directoryStoreFile           live.4.indexGroups                  reverseDirectoryStore.shadow
journals.live                       live.2.directoryStoreFile.shadow    live.4.indexHead                    reverseStore.updates
journals.repair                     live.2.indexArrays                  live.4.indexIds                     shutdown_time
journals.scan                       live.2.indexCompactDirectory        live.4.indexPositions               store.db
live.0.directoryStoreFile           live.2.indexDirectory               live.4.indexPostings                store.updates
live.0.directoryStoreFile.shadow    live.2.indexGroups                  live.4.indexUpdates                 store_generation
live.0.indexArrays                  live.2.indexHead                    live.4.shadowIndexGroups            tmp.Lion
live.0.indexCompactDirectory        live.2.indexIds                     live.4.shadowIndexHead              tmp.SnowLeopard
live.0.indexDirectory               live.2.indexPositions               live.5.directoryStoreFile           tmp.spotlight.loc
live.0.indexGroups                  live.2.indexPostings                live.5.directoryStoreFile.shadow    tmp.spotlight.state
live.0.indexHead                    live.2.shadowIndexGroups            live.5.indexArrays
```



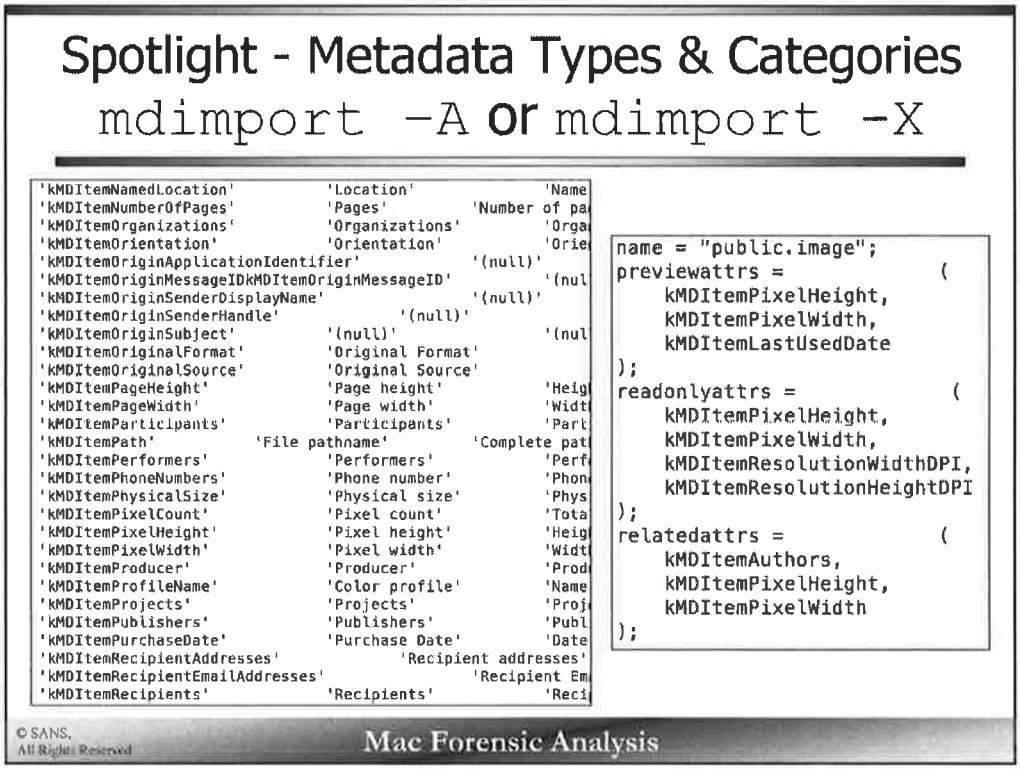
One directory that can be of use to forensic analysts is the Spotlight Cache directory. This directory contains a nesting of other sub-directories containing many text files. These text files hold text-based versions of their original documents including e-mail, documents, and chats.

The number preceding each .txt extension, is the CNID/inode number for the file it represents. The lower screenshot shows an example of this. The filename `375242.txt` contains the e-mail related data.

We can find the associated file by searching the system for the CNID, `375242`. We can use the `find` command on the mounted volume. Use the `-inum` argument to find the file by CNID/inode number.

```
sudo find /Volumes/dade_mounted/ -inum 375242
```

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



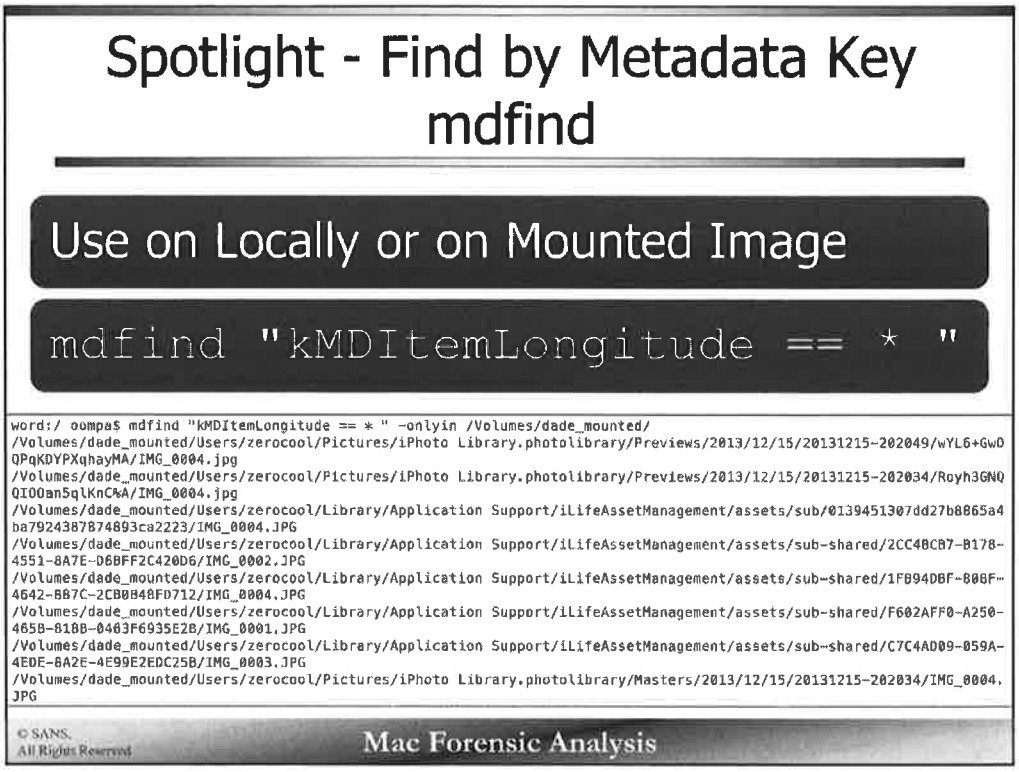
Two native command line utilities can show us what type of data may be indexed. These commands will only show the attributes that are associated with the host system, not the mounted system. Many of the `kMD*` attributes will be the same across systems, but keep an eye out for those application specific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

The `mdimport -A` command (shown on the left) will print out the attributes names and descriptions.

The `mdimport -X` command (shown on the right) prints out metadata attributes associated with a specific file type. For example, attributes associated with photos.

Reference:

Man page for `mdimport`

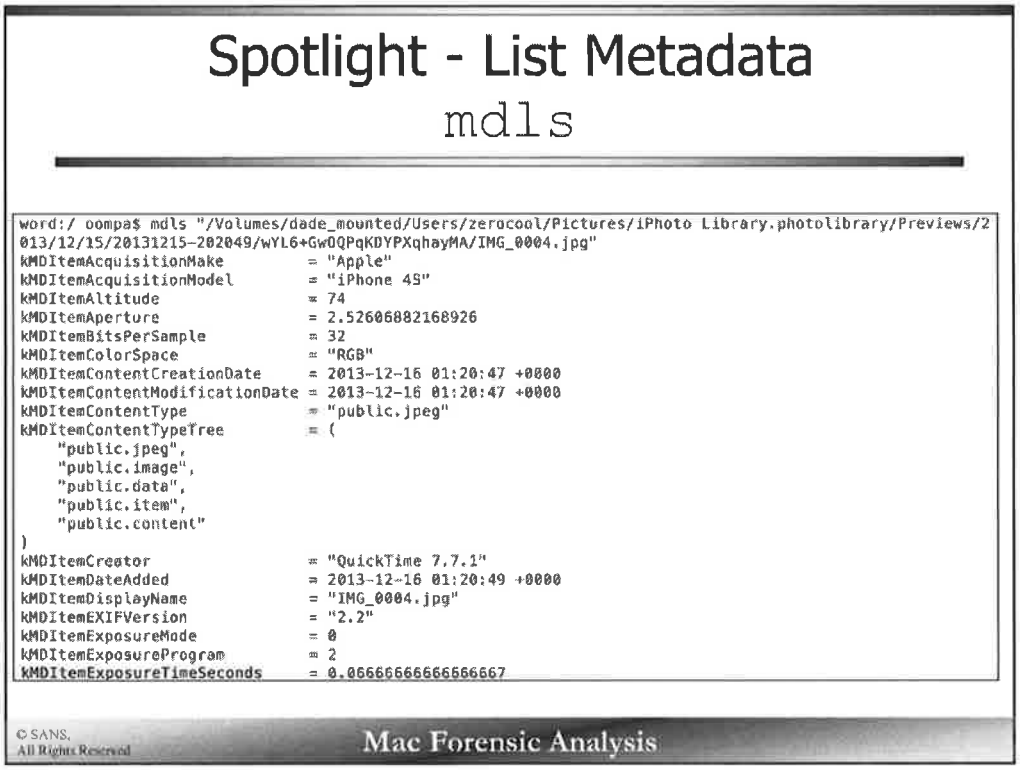


The `mdfind` command will find files based on certain metadata criteria.

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

Reference

Man page for `mdfind`



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

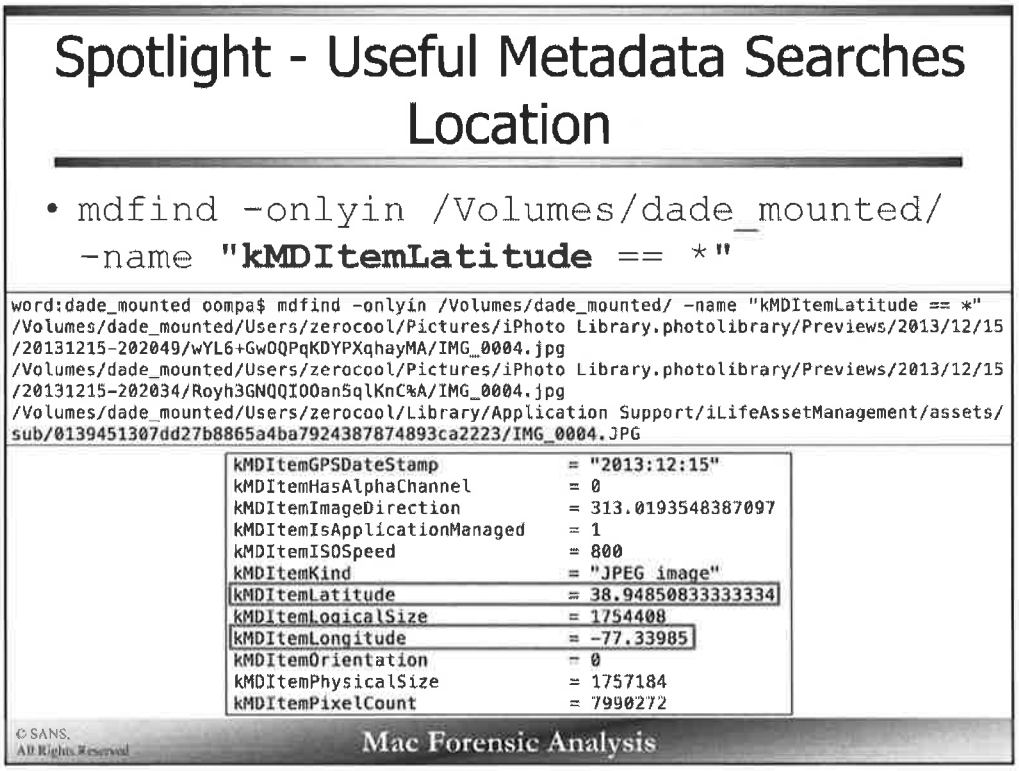
- Timestamps
- Content Types
- Download Date
- Download Location
- File Sizes
- File Ownership
- File Properties

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

Reference:

Man page for `mdls`

```
word:/ oompa$ mdls "/Volumes/dade_mounted/Users/zerocool/Pictures/iPhoto Library.photolibrary/Previews/2
013/12/15/20131215-202049/wYL6+GwOQPqKDYPXqhayMA/IMG_0004.jpg"
kMDItemAcquisitionMake          = "Apple"
kMDItemAcquisitionModel         = "iPhone 4S"
kMDItemAltitude                 = 74
kMDItemAperture                 = 2.52606882168926
kMDItemBitsPerSample            = 32
kMDItemColorSpace               = "RGB"
kMDItemContentCreationDate      = 2013-12-16 01:20:47 +0000
kMDItemContentModificationDate  = 2013-12-16 01:20:47 +0000
kMDItemContentType              = "public.jpeg"
kMDItemContentTypeTree          = (
    "public.jpeg",
    "public.image",
    "public.data",
    "public.item",
    "public.content"
)
kMDItemCreator                  = "QuickTime 7.7.1"
kMDItemDateAdded                = 2013-12-16 01:20:49 +0000
kMDItemDisplayName              = "IMG_0004.jpg"
kMDItemEXIFVersion              = "2.2"
kMDItemExposureMode             = 0
kMDItemExposureProgram          = 2
kMDItemExposureTimeSeconds      = 0.06666666666666667
kMDItemFlashOnOff               = 0
kMDItemFNumber                  = 2.4
kMDItemFocalLength              = 4.28
kMDItemFSContentChangeDate      = (null)
kMDItemFSCreationDate           = (null)
kMDItemFSCreatorCode            = ""
kMDItemFSFinderFlags            = (null)
kMDItemFSHasCustomIcon          = (null)
kMDItemFSInvisible              = 0
kMDItemFSIsExtensionHidden      = (null)
kMDItemFSIsStationery           = (null)
kMDItemFSLabel                  = 0
kMDItemFSName                   = (null)
kMDItemFSNodeCount              = (null)
kMDItemFSOwnerGroupID           = (null)
kMDItemFSOwnerUserID            = (null)
kMDItemFSSize                   = 1754408
kMDItemFSTypeCode               = ""
kMDItemGPSDateStamp             = "2013:12:15"
kMDItemHasAlphaChannel          = 0
kMDItemImageDirection           = 313.0193548387097
kMDItemIsApplicationManaged     = 1
kMDItemISOSpeed                 = 800
kMDItemKind                     = "JPEG image"
kMDItemLatitude                 = 38.94850833333334
kMDItemLogicalSize              = 1754408
kMDItemLongitude                = -77.33985
kMDItemOrientation              = 0
kMDItemPhysicalSize             = 1757184
kMDItemPixelCount               = 7990272
kMDItemPixelHeight              = 2448
kMDItemPixelWidth               = 3264
kMDItemProfileName              = "sRGB IEC61966-2.1"
kMDItemRedEyeOnOff              = 0
kMDItemResolutionHeightDPI      = 72
kMDItemResolutionWidthDPI       = 72
kMDItemSupportFileType          = (
    MDSystemFile
)
kMDItemTimestamp                = "00:59:06"
kMDItemWhiteBalance             = 0
```



We can use created command line searches to find and print out very specific data.

This example uses the `mdfind` command with the `-onlyin` argument to limit the search in the mounted image directory to search for files that have the `kMDItemLatitute` metadata attribute.

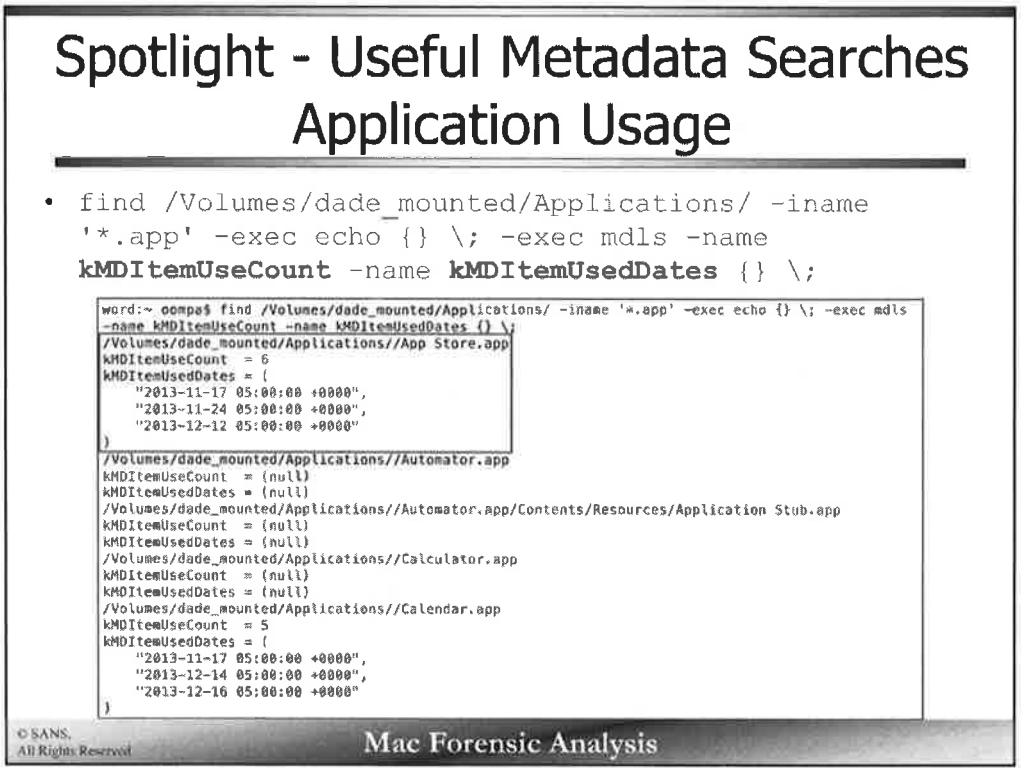
This will print out all files that contain locational attributes on the mounted image.

The bottom screenshot shows the output `mdls` on of one of the files with the coordinates that were indexed from this photo.

Reference:

Man page for `mdfind`

Man page for `mdls`



This command prints the name of an application along with the "Use Count" and "Item Used" timestamps associated with it. This can give a good idea of when an application was used, this is very similar to that of Windows Prefetch data.

One caveat to the `kMDItemUsedDates` is that these are only created once per a given day. Another attribute `kMDItemUseCount` will keep track of the number of times an application was used.

The `kMDItemUsedDates` timestamps show the day as well as the time zone of the application usage.

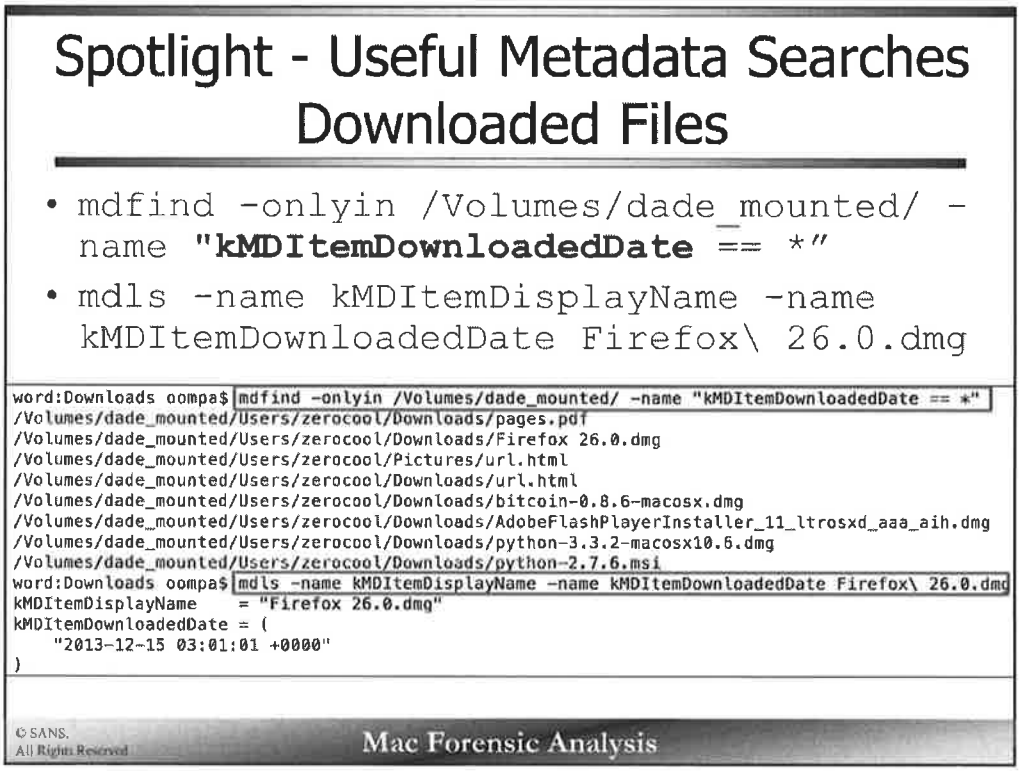
For example, App Store.app was used six times, on three days (all in UTC -5:00 time zone):

- 11/17/2013
- 11/24/2013
- 12/12/2013

The following command uses the `find` command on the `/Volumes/dade_mounted/Applications/` directory to search for files with the .app extension (`-iname`) and executes the following command (`-exec`).

```
echo {} \; -exec mdls -name kMDItemUsedDates {}
```

This part of the command `echo`'s the file path and name of the file found with the `find` command. (The curly brackets {} are used as a variable to store the filename/path.) The second `-exec`, prints the metadata output for only the `kKMItemUsedDates`.

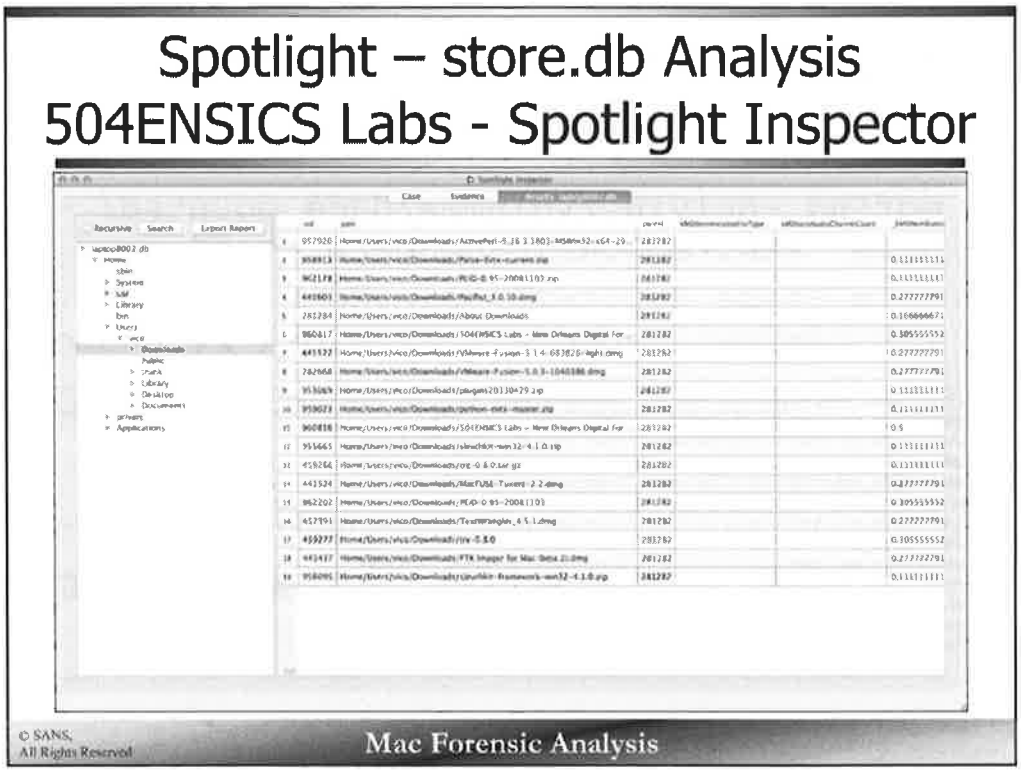


Wc can scarch for downloaded items by using the `kMDItemDownloadedDate` metadata attribute with the `mdfind` utility.

```
mdfind -onlyin /Volumes/dade_mounted/ -name "kMDItemDownloadedDate == *"
```

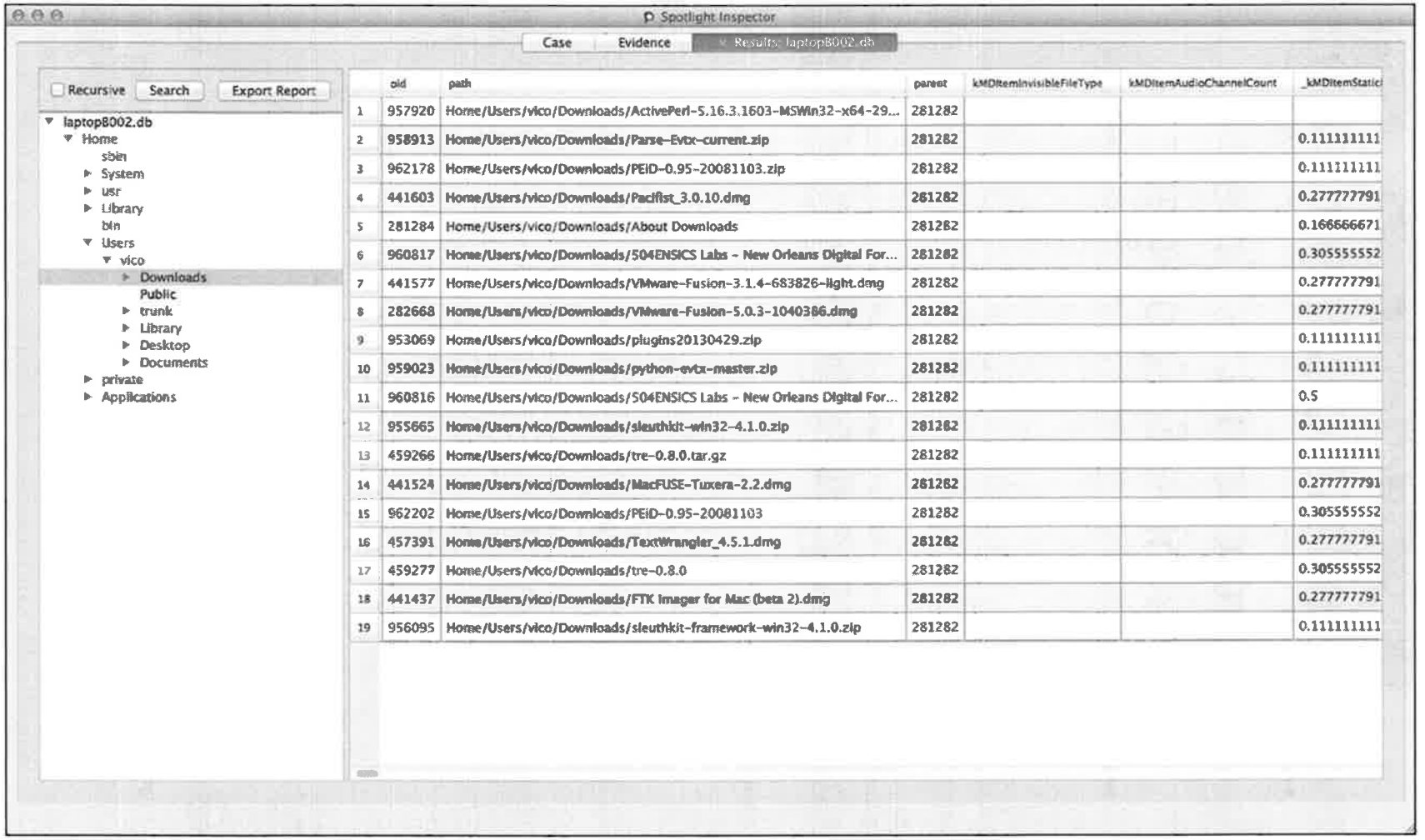
If we use the `mdls` command on one of these files to show the name and when the item was downloaded.

```
mdls -name kMDItemDisplayName -name kMDItemDownloadedDate Firefox\ 26.0.dmg
```

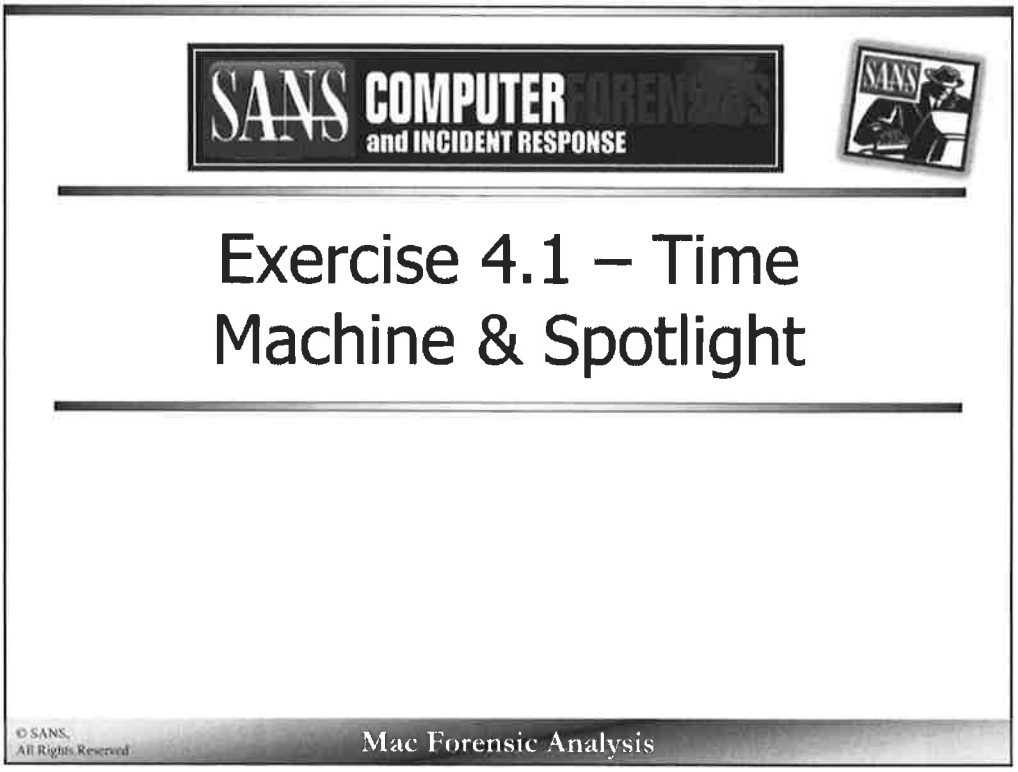


504ENSICS Labs has created a tool, Spotlight Inspector, to analyze the `store.db` Spotlight databases. This tool is available at `http://www.504ensics.com/tools/digital-forensics-tool-spotlight-inspector/`.

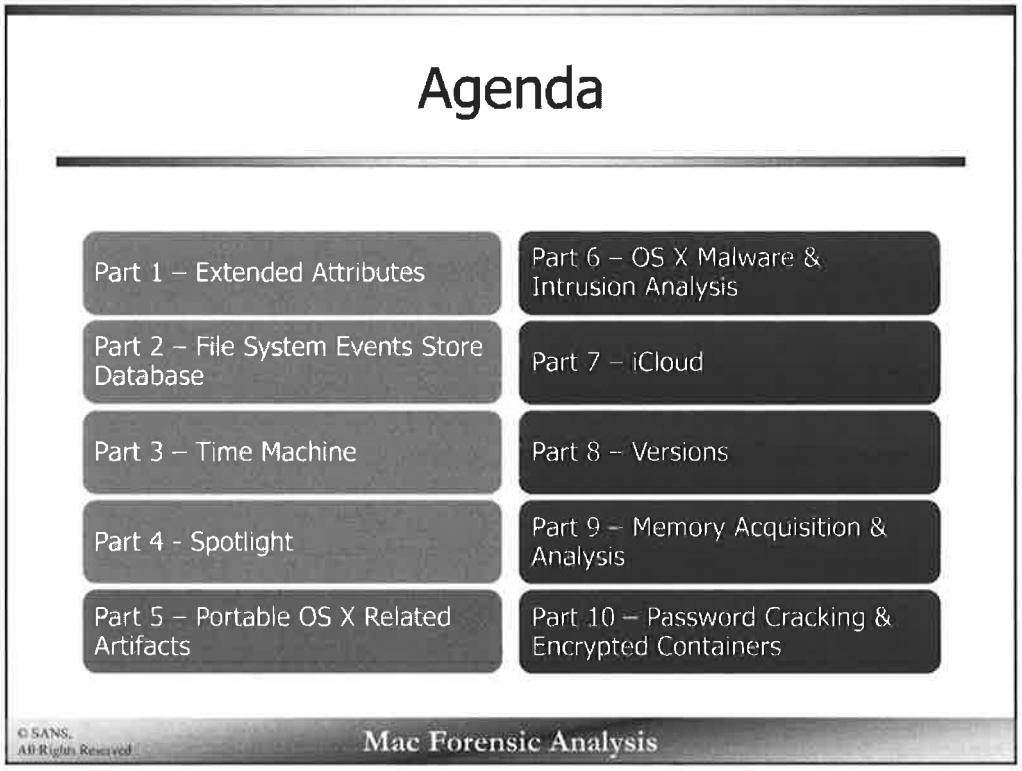
This tool will show all the various metadata attributes associated with the files. This tool can also do timeline and differential analysis.



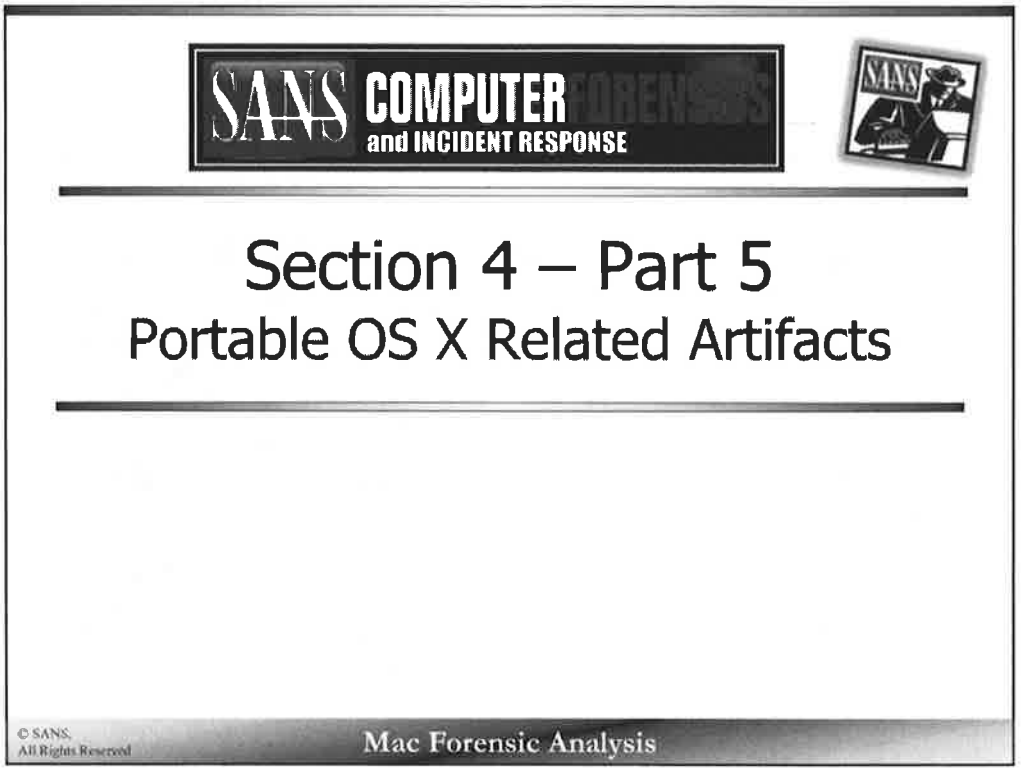
	oid	path	parent	kMDItemInvisibleFileType	kMDItemAudioChannelCount	_kMDItemStatic
1	957920	Home/Users/vico/Downloads/ActivePerl-5.16.3.1603-MSWin32-x64-29...	281282			
2	958913	Home/Users/vico/Downloads/Parse-Evtx-current.zip	281282			0.111111111
3	962178	Home/Users/vico/Downloads/PEiD-0.95-20081103.zip	281282			0.111111111
4	441603	Home/Users/vico/Downloads/Pacifist_3.0.10.dmg	281282			0.277777791
5	281284	Home/Users/vico/Downloads/About Downloads	281282			0.166666671
6	960817	Home/Users/vico/Downloads/504ENSICS Labs - New Orleans Digital For...	281282			0.305555552
7	441577	Home/Users/vico/Downloads/VMware-Fusion-3.1.4-683826-light.dmg	281282			0.277777791
8	282668	Home/Users/vico/Downloads/VMware-Fusion-5.0.3-1040386.dmg	281282			0.277777791
9	953069	Home/Users/vico/Downloads/plugins20130429.zip	281282			0.111111111
10	959023	Home/Users/vico/Downloads/python-evtx-master.zip	281282			0.111111111
11	960816	Home/Users/vico/Downloads/504ENSICS Labs - New Orleans Digital For...	281282			0.5
12	955665	Home/Users/vico/Downloads/sleuthkit-win32-4.1.0.zip	281282			0.111111111
13	459266	Home/Users/vico/Downloads/tre-0.8.0.tar.gz	281282			0.111111111
14	441524	Home/Users/vico/Downloads/MacFUSE-Tuxera-2.2.dmg	281282			0.277777791
15	962202	Home/Users/vico/Downloads/PEiD-0.95-20081103	281282			0.305555552
16	457391	Home/Users/vico/Downloads/TextWrangler_4.5.1.dmg	281282			0.277777791
17	459277	Home/Users/vico/Downloads/tre-0.8.0	281282			0.305555552
18	441437	Home/Users/vico/Downloads/FTK Imager for Mac (beta 2).dmg	281282			0.277777791
19	956095	Home/Users/vico/Downloads/sleuthkit-framework-win32-4.1.0.zip	281282			0.111111111



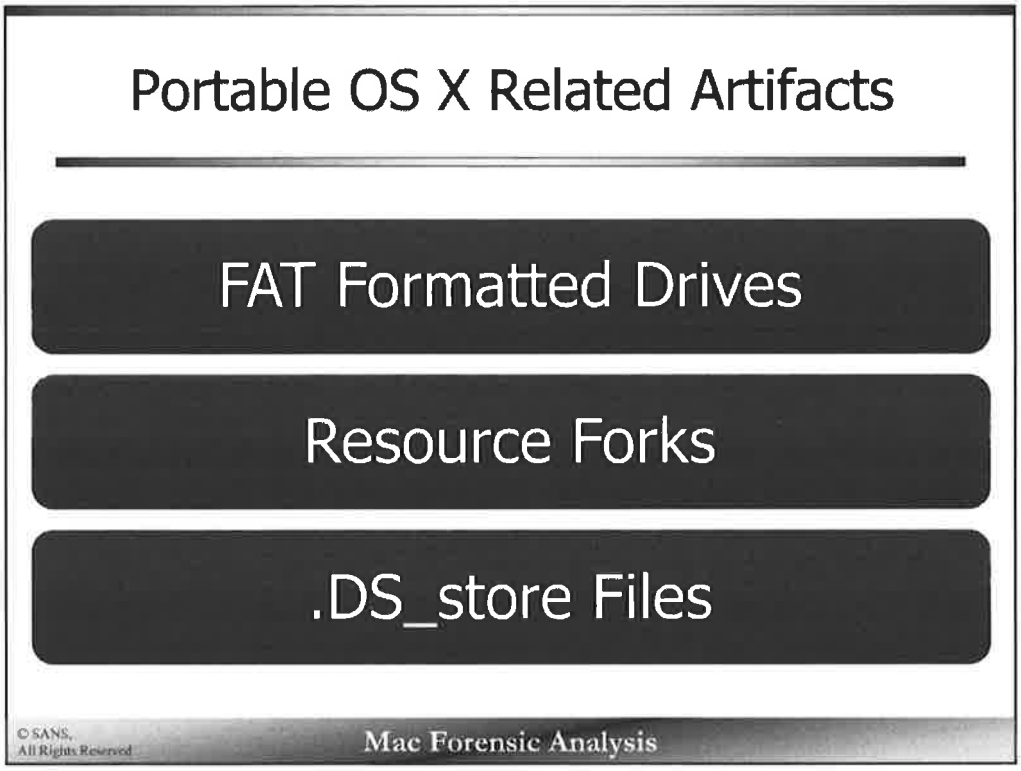
This page intentionally left blank.



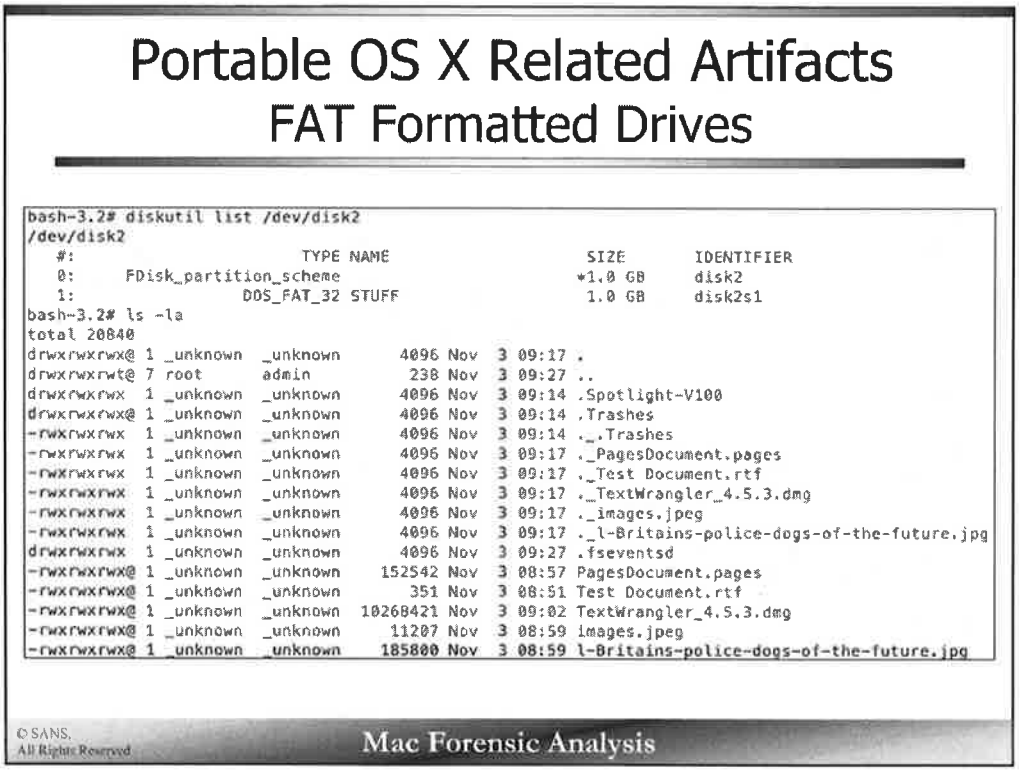
This page intentionally left blank.



This page intentionally left blank.



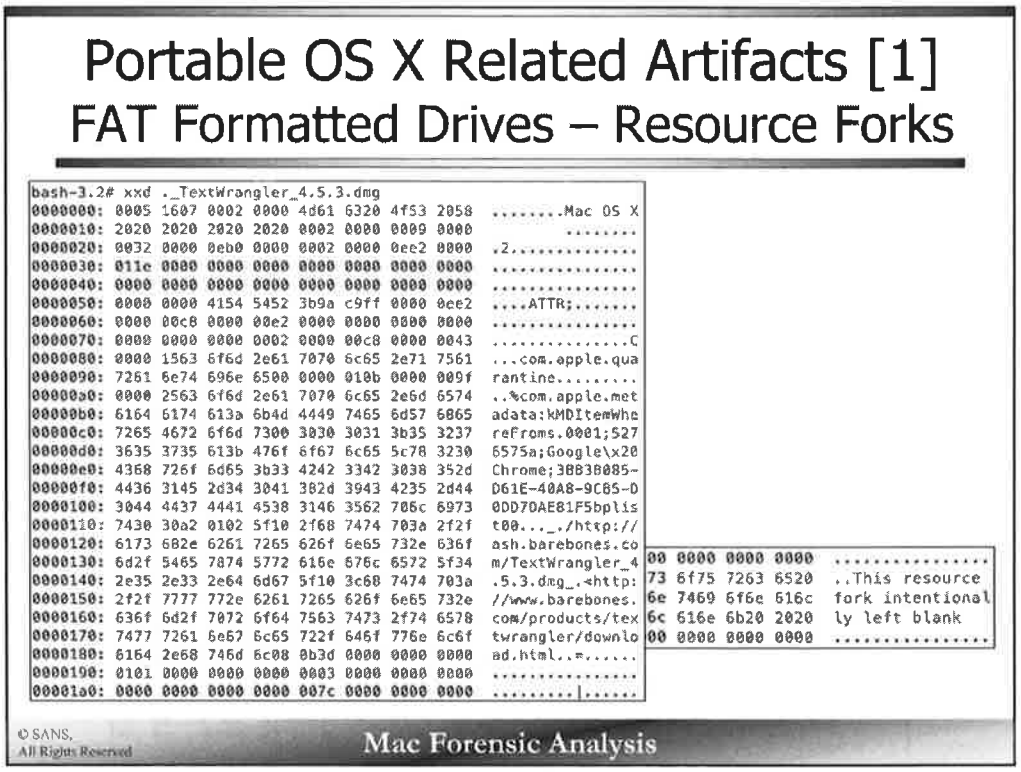
OS X can leave behind some interesting artifacts that may be shown on other non-OS X systems. These artifacts can help us determine if a file was originally found on a Mac system. These artifacts include Resource Fork data and those sneaky `.DS_store` files that seem to get everywhere!



Files copied to FAT or ExFAT formatted drives will show signs of OS X originations.

- Resource Forks (Extended Attributes)
- Spotlight Index
- Document Versions
- Trashes Directory
- .fseventsd Directory
- ./DocumentRevisions-
- …other additional hidden files/directories

```
bash-3.2# diskutil list /dev/disk2
/dev/disk2
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:     FDisk_partition_scheme                        *1.0 GB     disk2
   1:                 DOS_FAT_32 STUFF                   1.0 GB     disk2s1
bash-3.2# ls -la
total 20840
drwxrwxrwx@ 1 _unknown  _unknown      4096 Nov  3 09:17 .
drwxrwxrwt@ 7 root      admin          238 Nov  3 09:27 ..
drwxrwxrwx  1 _unknown  _unknown      4096 Nov  3 09:14 .Spotlight-V100
drwxrwxrwx@ 1 _unknown  _unknown      4096 Nov  3 09:14 .Trashes
-rwxrwxrwx  1 _unknown  _unknown      4096 Nov  3 09:14 ._.Trashes
-rwxrwxrwx  1 _unknown  _unknown      4096 Nov  3 09:17 ._PagesDocument.pages
-rwxrwxrwx  1 _unknown  _unknown      4096 Nov  3 09:17 ._Test Document.rtf
-rwxrwxrwx  1 _unknown  _unknown      4096 Nov  3 09:17 ._TextWrangler_4.5.3.dmg
-rwxrwxrwx  1 _unknown  _unknown      4096 Nov  3 09:17 ._images.jpeg
-rwxrwxrwx  1 _unknown  _unknown      4096 Nov  3 09:17 ._l-Britains-police-dogs-of-the-future.jpg
drwxrwxrwx  1 _unknown  _unknown      4096 Nov  3 09:27 .fseventsd
-rwxrwxrwx@ 1 _unknown  _unknown    152542 Nov  3 08:57 PagesDocument.pages
-rwxrwxrwx@ 1 _unknown  _unknown       351 Nov  3 08:51 Test Document.rtf
-rwxrwxrwx@ 1 _unknown  _unknown  10268421 Nov  3 09:02 TextWrangler_4.5.3.dmg
-rwxrwxrwx@ 1 _unknown  _unknown     11207 Nov  3 08:59 images.jpeg
-rwxrwxrwx@ 1 _unknown  _unknown    185800 Nov  3 08:59 l-Britains-police-dogs-of-the-future.jpg
```



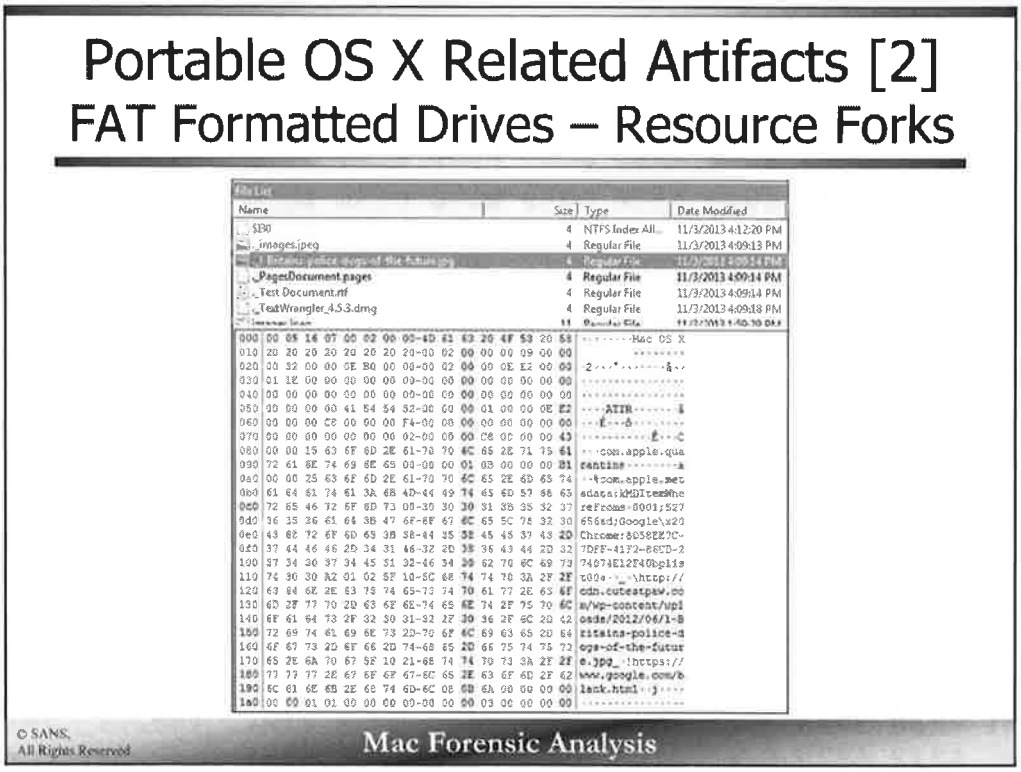
Extended Attributes that are normally stored in the $Attributes file on HFS+ file systems are stored in separate files on FAT32 and ExFAT file systems. These files share the same filename as the associated file with a "dot underscore" (. _) appended to the beginning. These files are 4,096 bytes in size.

Shown in the larger screenshot are the file signatures "`Mac OS X`" and "`ATTR;`" followed by the contents of the extended attribute. At the end of the file is the text "This resource fork intentionally left blank".

Warning Note: These files are copied over when full directories are copied, not necessarily if a single file is copied.

```
bash-3.2# xxd ._TextWrangler_4.5.3.dmg
0000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
0000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
0000020: 0032 0000 0eb0 0000 0002 0000 0ee2 0000  .2..............
0000030: 011e 0000 0000 0000 0000 0000 0000 0000  ................
0000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000050: 0000 0000 4154 5452 3b9a c9ff 0000 0ee2  ....ATTR;.......
0000060: 0000 00c8 0000 00e2 0000 0000 0000 0000  ................
0000070: 0000 0000 0000 0002 0000 00c8 0000 0043  ...............C
0000080: 0000 1563 6f6d 2e61 7070 6c65 2e71 7561  ...com.apple.qua
0000090: 7261 6e74 696e 6500 0000 010b 0000 009f  rantine.........
00000a0: 0000 2563 6f6d 2e61 7070 6c65 2e6d 6574  ..%com.apple.met
00000b0: 6164 6174 613a 6b4d 4449 7465 6d57 6865  adata:kMDItemWhe
00000c0: 7265 4672 6f6d 7300 3030 3031 3b35 3237  reFroms.0001;527
00000d0: 3635 3735 613b 476f 6f67 6c65 5c78 3230  6575a;Google\x20
00000e0: 4368 726f 6d65 3b33 4242 3342 3038 352d  Chrome;3BB3B085-
00000f0: 4436 3145 2d34 3041 382d 3943 4235 2d44  D61E-40A8-9CB5-D
0000100: 3044 4437 4441 4538 3146 3562 706c 6973  0DD7DAE81F5bplis
0000110: 7430 30a2 0102 5f10 2f68 7474 703a 2f2f  t00..._./http://
0000120: 6173 682e 6261 7265 626f 6e65 732e 636f  ash.barebones.co
0000130: 6d2f 5465 7874 5772 616e 676c 6572 5f34  m/TextWrangler_4
0000140: 2e35 2e33 2e64 6d67 5f10 3c68 7474 703a  .5.3.dmg_.<http:
0000150: 2f2f 7777 772e 6261 7265 626f 6e65 732e  //www.barebones.
0000160: 636f 6d2f 7072 6f64 7563 7473 2f74 6578  com/products/tex
0000170: 7477 7261 6e67 6c65 722f 646f 776e 6c6f  twrangler/downlo
0000180: 6164 2e68 746d 6c08 0b3d 0000 0000 0000  ad.html..=......
0000190: 0101 0000 0000 0000 0003 0000 0000 0000  ................
00001a0: 0000 0000 0000 0000 007c 0000 0000 0000  .........|......
```

```
0000ee0: 0000 0000 0100 0000 0100 0000 0000 0000  ................
0000ef0: 001e 5468 6973 2072 6573 6f75 7263 6520  ..This resource
0000f00: 666f 726b 2069 6e74 656e 7469 6f6e 616c  fork intentional
0000f10: 6c79 206c 6566 7420 626c 616e 6b20 2020  ly left blank
0000f20: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

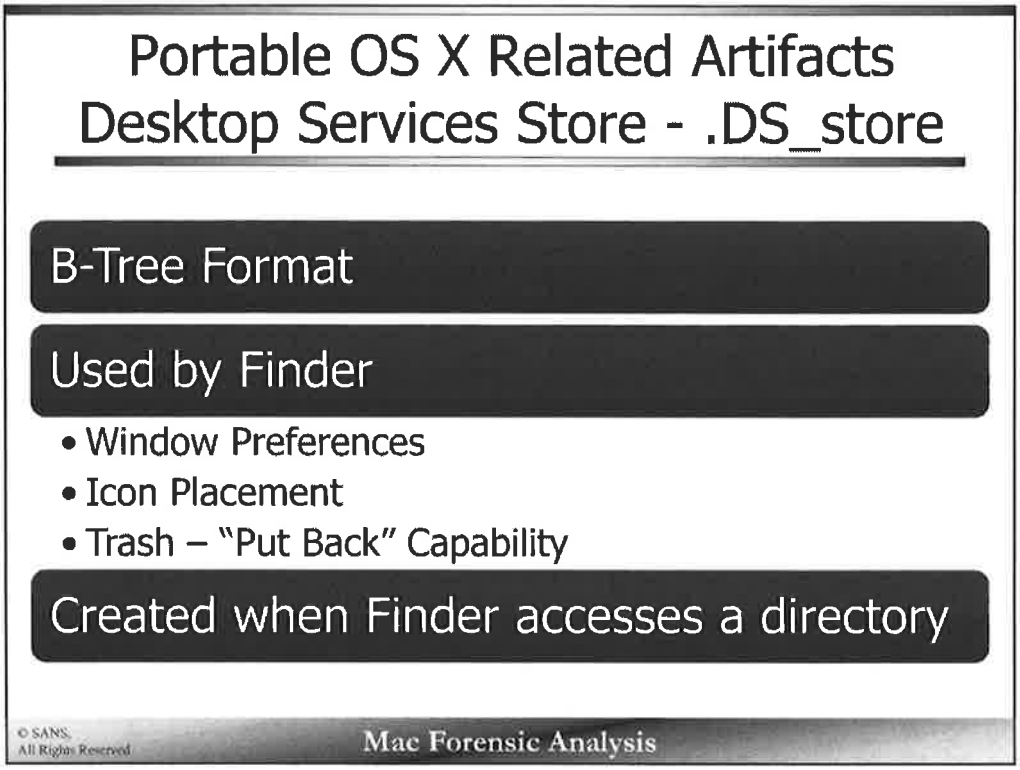


These files are hidden, but using forensic software, shown here is FTK Imager Lite, you can easily view these files.

File List

Name	Size	Type	Date Modified
$I30	4	NTFS Index All...	11/3/2013 4:12:20 PM
._images.jpeg	4	Regular File	11/3/2013 4:09:13 PM
._1-Britains-police-dogs-of-the-future.jpg	4	Regular File	11/3/2013 4:09:14 PM
._PagesDocument.pages	4	Regular File	11/3/2013 4:09:14 PM
._Test Document.rtf	4	Regular File	11/3/2013 4:09:14 PM
._TextWrangler_4.5.3.dmg	4	Regular File	11/3/2013 4:09:18 PM
images.jpeg	11	Regular File	11/3/2013 1:59:20 PM

```
000 00 05 16 07 00 02 00 00-4D 61 63 20 4F 53 20 58 ........Mac OS X
010 20 20 20 20 20 20 20 20-00 02 00 00 00 09 00 00          ........
020 00 32 00 00 0E B0 00 00-00 02 00 00 0E E2 00 00 .2...°.......â..
030 01 1E 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
050 00 00 00 00 41 54 54 52-00 00 00 01 00 00 0E E2 ....ATTR.......â
060 00 00 00 C8 00 00 00 F4-00 00 00 00 00 00 00 00 ...È...ô........
070 00 00 00 00 00 00 00 02-00 00 00 C8 00 00 00 43 ...........È...C
080 00 00 15 63 6F 6D 2E 61-70 70 6C 65 2E 71 75 61 ...com.apple.qua
090 72 61 6E 74 69 6E 65 00-00 00 01 0B 00 00 00 B1 rantine........±
0a0 00 00 25 63 6F 6D 2E 61-70 70 6C 65 2E 6D 65 74 ..%com.apple.met
0b0 61 64 61 74 61 3A 6B 4D-44 49 74 65 6D 57 68 65 adata:kMDItemWhe
0c0 72 65 46 72 6F 6D 73 00-30 30 30 31 3B 35 32 37 reFroms.0001;527
0d0 36 35 36 61 64 3B 47 6F-6F 67 6C 65 5C 78 32 30 656ad;Google\x20
0e0 43 68 72 6F 6D 65 3B 38-44 35 38 45 45 37 43 2D Chrome;8D58EE7C-
0f0 37 44 46 46 2D 34 31 46-32 2D 38 36 43 44 2D 32 7DFF-41F2-86CD-2
100 37 34 30 37 34 45 31 32-46 34 30 62 70 6C 69 73 74074E12F40bplis
110 74 30 30 A2 01 02 5F 10-5C 68 74 74 70 3A 2F 2F t00¢..._.\http://
120 63 64 6E 2E 63 75 74 65-73 74 70 61 77 2E 63 6F cdn.cutestpaw.co
130 6D 2F 77 70 2D 63 6F 6E-74 65 6E 74 2F 75 70 6C m/wp-content/upl
140 6F 61 64 73 2F 32 30 31-32 2F 30 36 2F 6C 2D 42 oads/2012/06/1-B
150 72 69 74 61 69 6E 73 2D-70 6F 6C 69 63 65 2D 64 ritains-police-d
160 6F 67 73 2D 6F 66 2D 74-68 65 2D 66 75 74 75 72 ogs-of-the-futur
170 65 2E 6A 70 67 5F 10 21-68 74 74 70 73 3A 2F 2F e.jpg_.!https://
180 77 77 77 2E 67 6F 6F 67-6C 65 2E 63 6F 6D 2F 62 www.google.com/b
190 6C 61 6E 6B 2E 68 74 6D-6C 08 0B 6A 00 00 00 00 lank.html..j....
1a0 00 00 01 01 00 00 00 00-00 00 00 03 00 00 00 00 ................
```



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

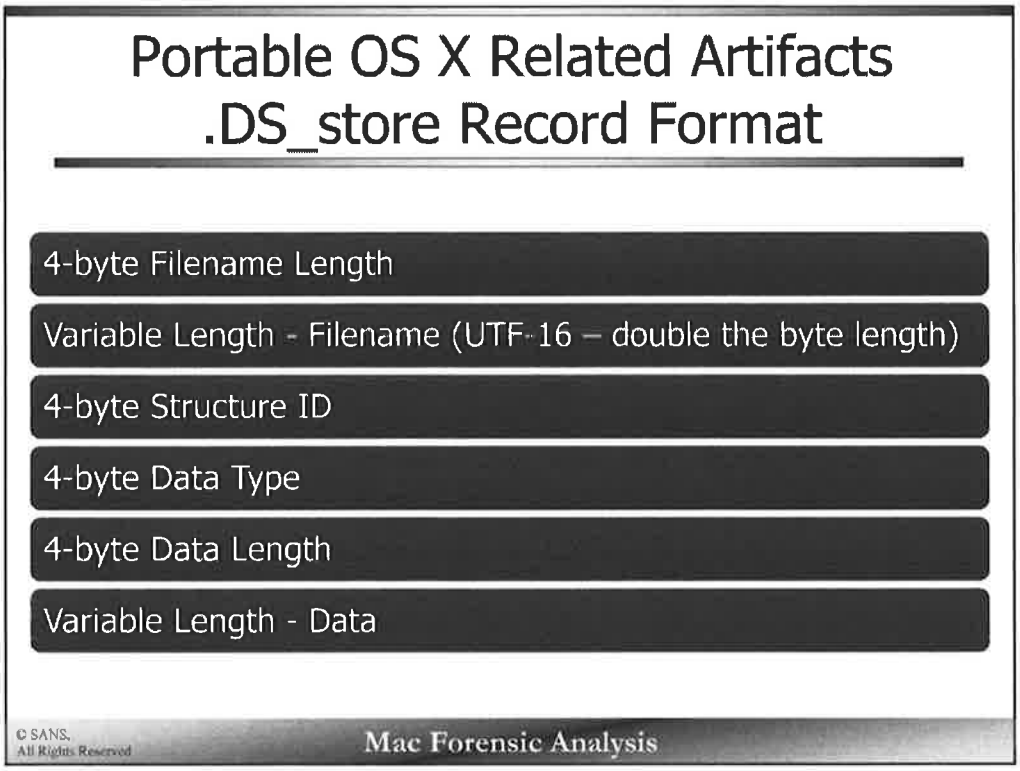
These files are created when the Finder application is used to view a directory. While all over OS X drives, these files are copied along with files to external hard drives, thumb drives, or network drives. These files implement a B-tree format and are used to save the Finder viewing settings. Settings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

References:

http://search.cpan.org/~wiml/Mac-Finder-DSStore-0.95/DSStoreFormat.pod

https://wiki.mozilla.org/DS_Store_File_Format

https://github.com/dscho/dsstore



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

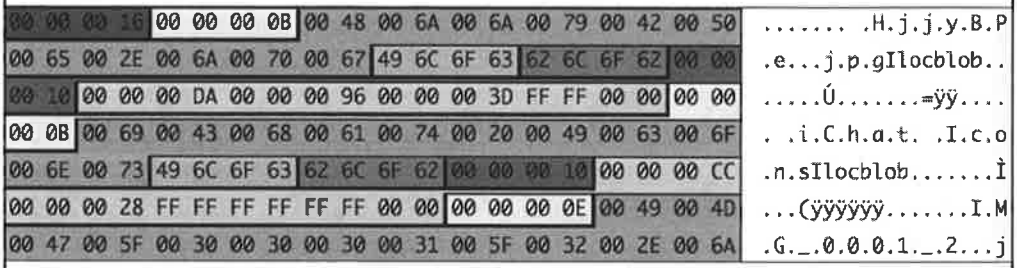
References:

http://search.cpan.org/~wiml/Mac-Finder-DSStore-0.95/DSStoreFormat.pod

https://wiki.mozilla.org/DS_Store_File_Format

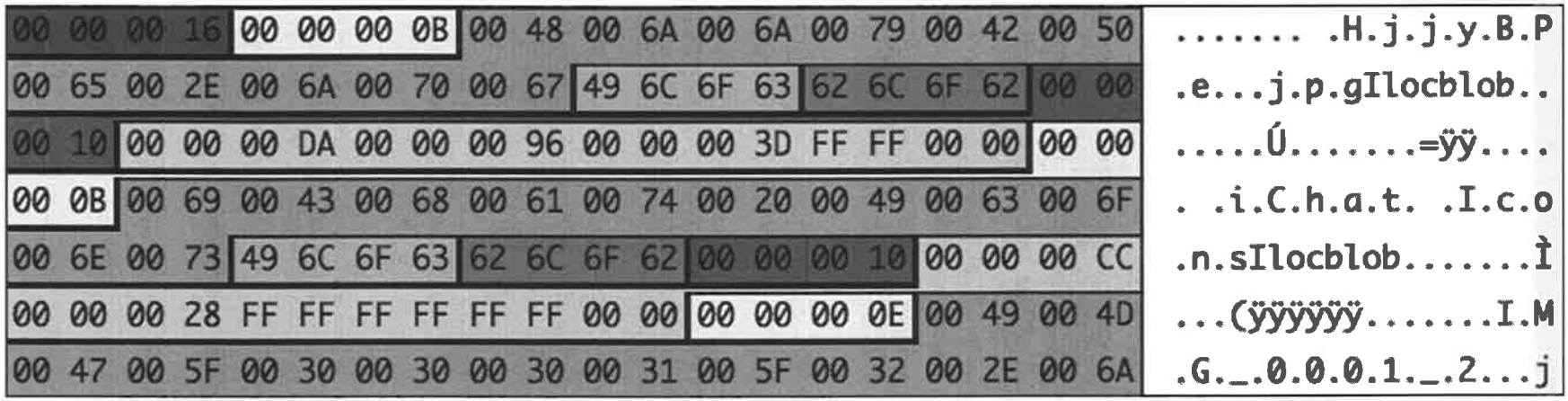
https://github.com/dscho/dsstore

Portable OS X Related Artifacts
.DS_store Record Format - Example



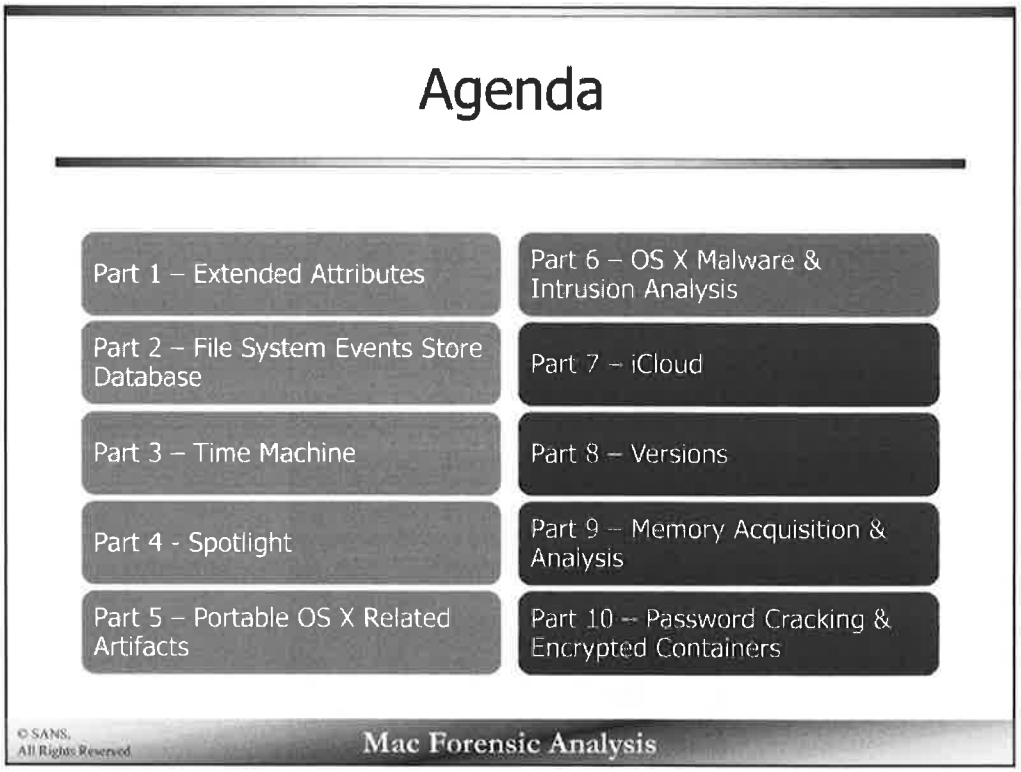
```
00 00 00 16 00 00 00 0B 00 48 00 6A 00 6A 00 79 00 42 00 50   ....... .H.j.j.y.B.P
00 65 00 2E 00 6A 00 70 00 67 49 6C 6F 63 62 6C 6F 62 00 00   .e...j.p.gIlocblob..
00 10 00 00 00 DA 00 00 00 96 00 00 00 3D FF FF 00 00 00 00   .....Ú.......=ÿÿ....
00 0B 00 69 00 43 00 68 00 61 00 74 00 20 00 49 00 63 00 6F   . .i.C.h.a.t. .I.c.o
00 6E 00 73 49 6C 6F 63 62 6C 6F 62 00 00 00 10 00 00 00 CC   .n.sIlocblob.......Ì
00 00 00 28 FF FF FF FF FF FF 00 00 00 00 00 0E 00 49 00 4D   ...(ÿÿÿÿÿÿ.......I.M
00 47 00 5F 00 30 00 30 00 30 00 31 00 5F 00 32 00 2E 00 6A   .G._.0.0.0.1._.2...j
```


Field	Size (bytes)	Data
Number of Records	4	0x00000016 = 22 Records
Record 1 – Filename Size	4	0x0000000B = 11 bytes
Record 1 - Filename	Variable	"HjjtBPe.jpg" (UTF-16 – length doubled)
Record 1 – Structure ID	4	"Iloc" (Icon Location)
Record 1 – Data Type	4	"blob"
Record 1 – Data Size	4	0x00000010 = 16
Record 1 - Data	Variable	0x000000DA00000096000003DFFFF0000
Record 2 – Filename Size	4	0x0000000B = 11 bytes
Record 2 - Filename	Variable	"iChat Icons" (UTF-16 – length doubled)
Record 2 – Structure ID	4	"Iloc" (Icon Location)
Record 2 – Data Type	4	"blob"
Record 2 – Data Size	4	0x00000010 = 16
Record 2 - Data	Variable	0x000000CC00000028FFFFFFFFFFFF0000

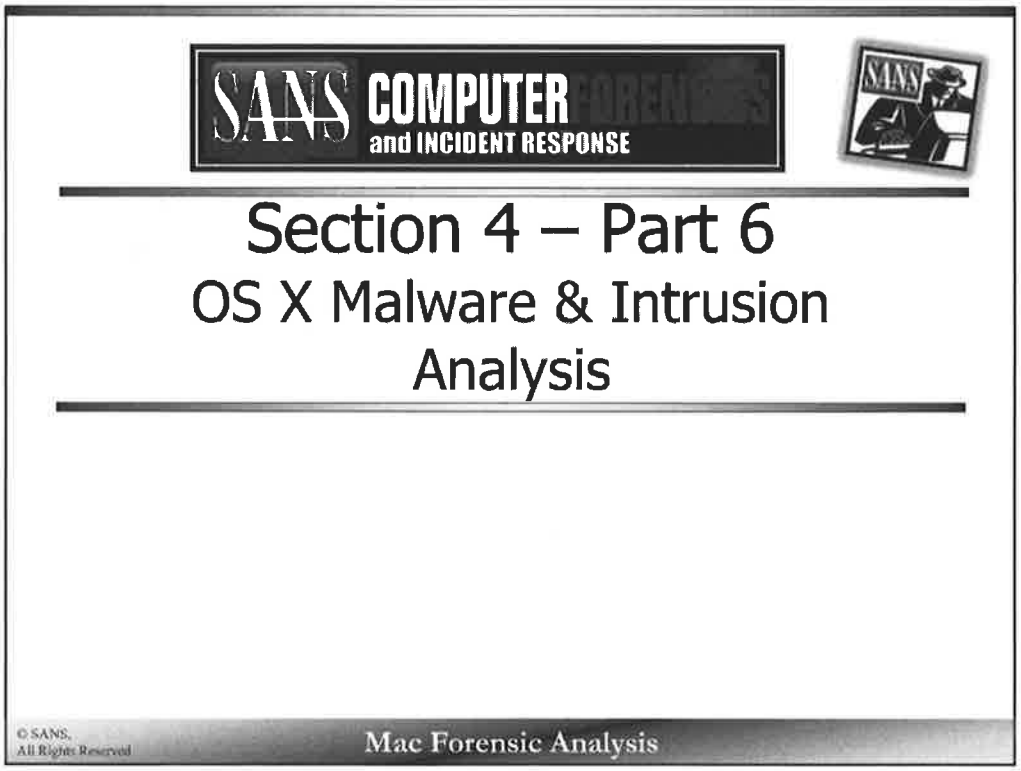


00	00	00	16	00	00	00	0B	00	48	00	6A	00	6A	00	79	00	42	00	50
00	65	00	2E	00	6A	00	70	00	67	49	6C	6F	63	62	6C	6F	62	00	00
00	10	00	00	00	DA	00	00	00	96	00	00	00	3D	FF	FF	00	00	00	00
00	0B	00	69	00	43	00	68	00	61	00	74	00	20	00	49	00	63	00	6F
00	6E	00	73	49	6C	6F	63	62	6C	6F	62	00	00	00	10	00	00	00	CC
00	00	00	28	FF	FF	FF	FF	FF	FF	00	00	00	00	00	0E	00	49	00	4D
00	47	00	5F	00	30	00	30	00	30	00	31	00	5F	00	32	00	2E	00	6A

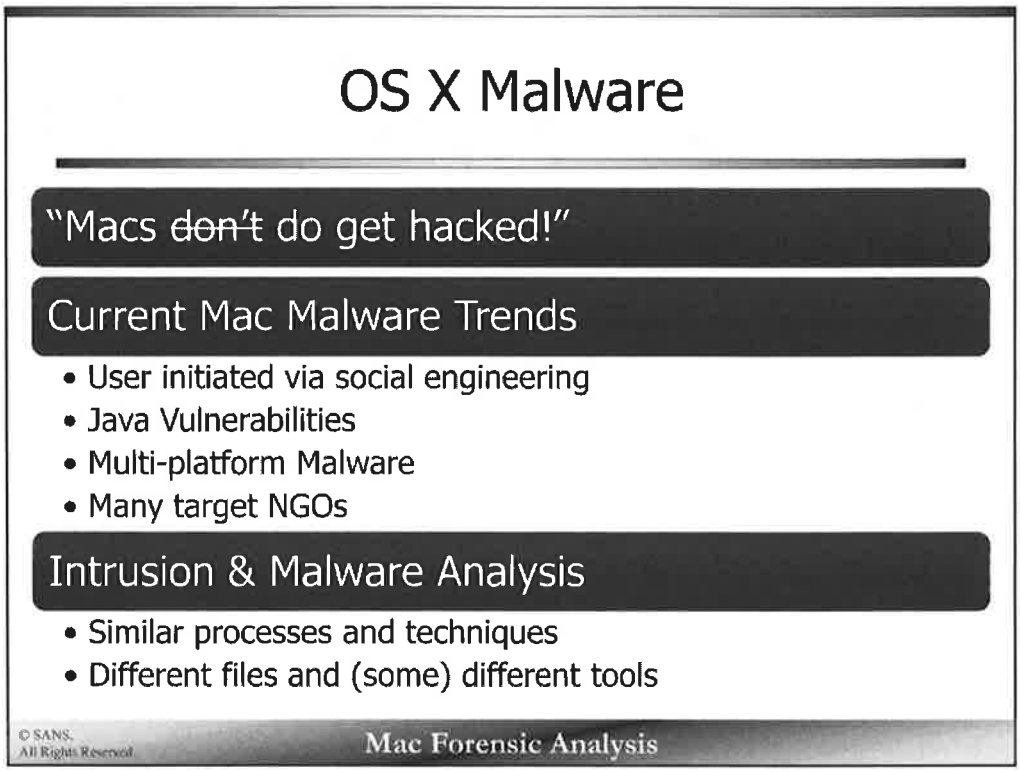
..... .H.j.j.y.B.P
.e...j.p.gIlocblob..
.....Ú.....=ÿÿ....
. .i.C.h.a.t. .I.c.o
.n.sIlocblob.....Ì
...(ÿÿÿÿÿÿ.....I.M
.G._.0.0.0.1._.2...j



This page intentionally left blank.



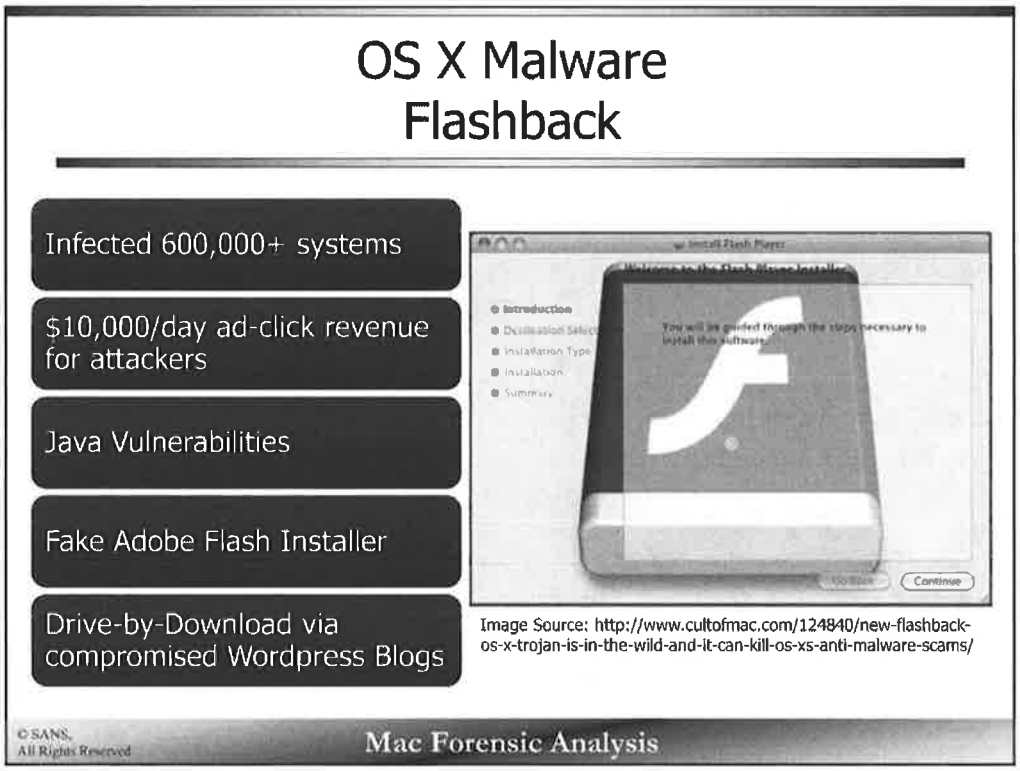
This page intentionally left blank.



Macs do in fact get hacked (every once in a while). Their popularity is opening up a whole new market for attackers. Java vulnerabilities in particular have created a market that is creating multi-platform malware. Why target just Windows users, when everyone who uses Java is vulnerable?

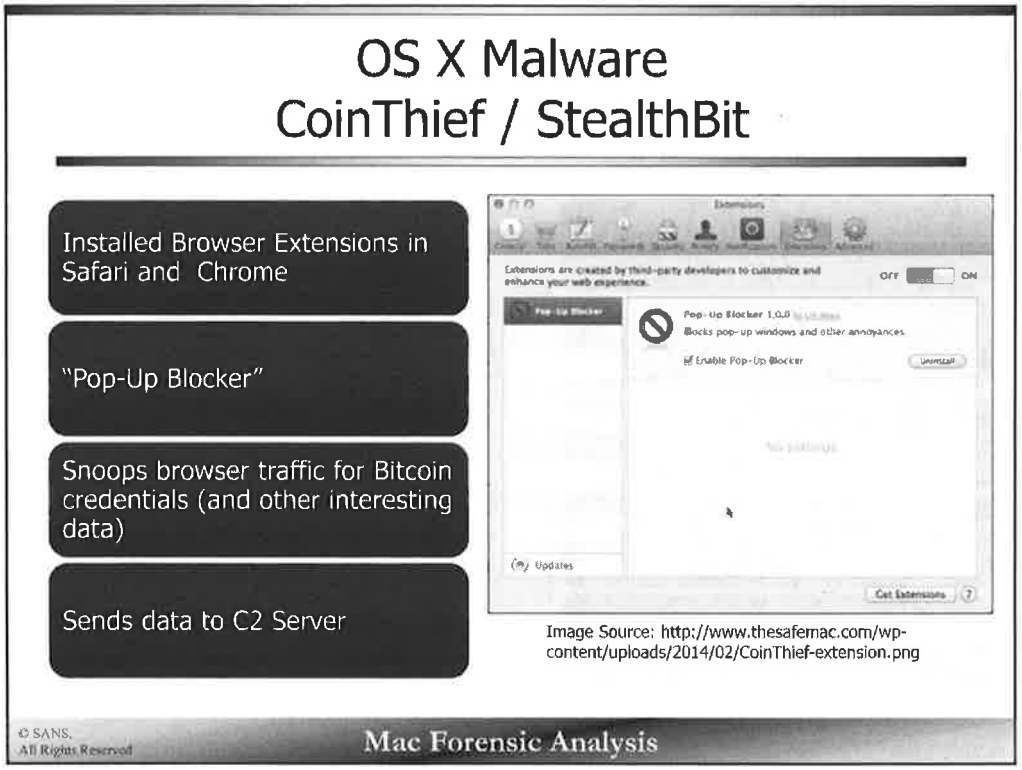
Mac malware on the whole is not as advanced as Windows malware, but frankly, it still gets the job done! A user will click on just about anything given the right phishing e-mail. Most of the Mac malware currently out there needs to be initiated by the user so the vector tends to be an attachment via an e-mail or a link to a malicious website.

Analysis of a Mac malware intrusion is very similar to that of a Windows intrusion – the techniques and processes are similar but the files and tools may be different.



The Flashback malware infected more than 600,000 systems. In terms of Mac infections this was HUGE! The Windows CodeRed infection rate was ~400,000. (Compared to current Windows infections, it's a drop in the bucket – but you get my point.)

This malware was used to generate ad-click revenue for the attackers and was delivered by a Java vulnerability that showed up as a fake Adobe Flash installer.

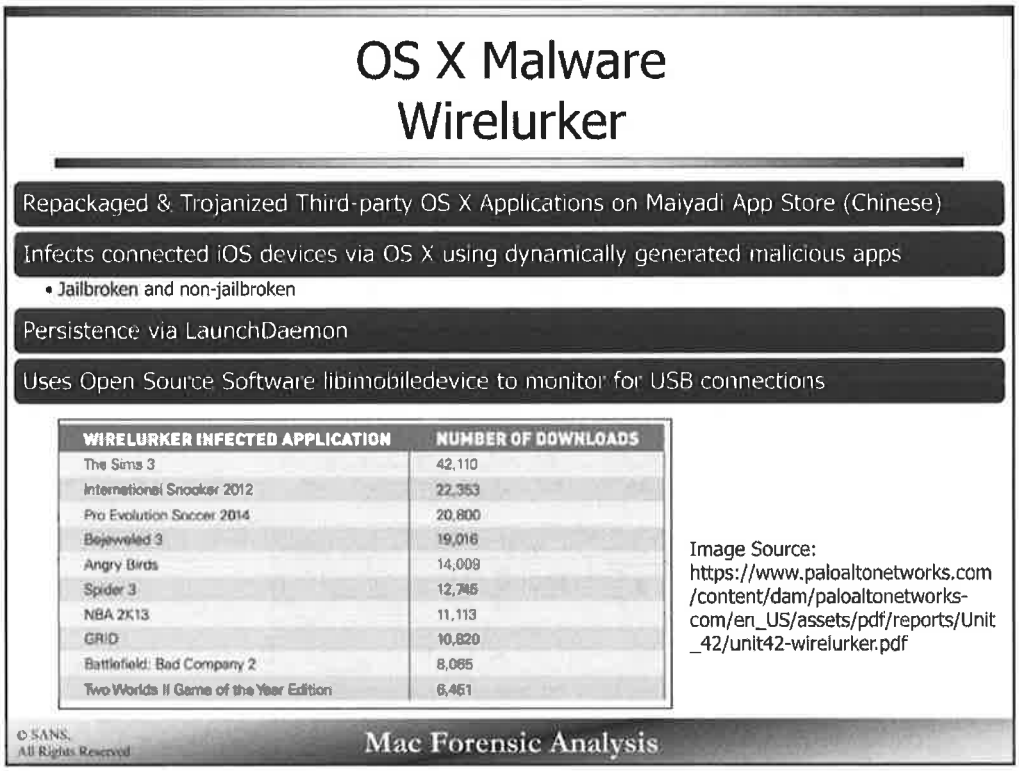


The CoinThief/StealthBit malware installed a browser extension to capture credentials to specific Bitcoin-related websites. These credentials were then sent to a command and control server.

References:

http://readwrite.com/2014/02/10/stealthbit-mac-osx-trojan-malware-steals-bitcoins

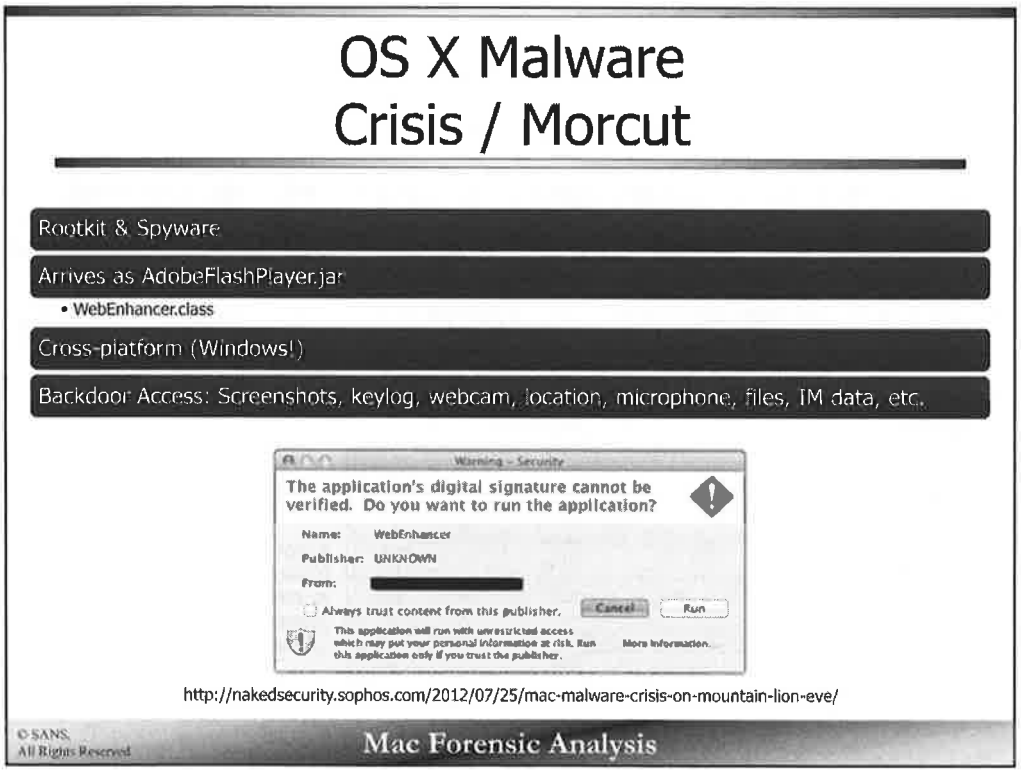
http://www.securemac.com/CoinThief-BitCoin-Trojan-Horse-MacOSX.php



Wirelurker is significant because it uses infected OS X systems to further infect iOS devices. Both jailbroken and non-jailbroken methods are used to infects the iDevices. OS X systems are initially infected by users downloading trojanized third-party applications through Chinese app stores.

References:

https://www.paloaltonetworks.com/content/dam/paloaltonetworks-com/en_US/assets/pdf/reports/Unit_42/unit42-wirelurker.pdf



The Crisis/Morcut malware was interesting because it is cross-platform. This malware arrives via Java as another fake Adobe Flash Player software.

This software creates backdoor access on the system and is reported to get access to take screenshots, install a keylogger, and access the webcam, locational data, microphone, files, and instant messenger data.

One of the malicious Java `.class` files for this malware is named `WebEnhancer.class`.

References:

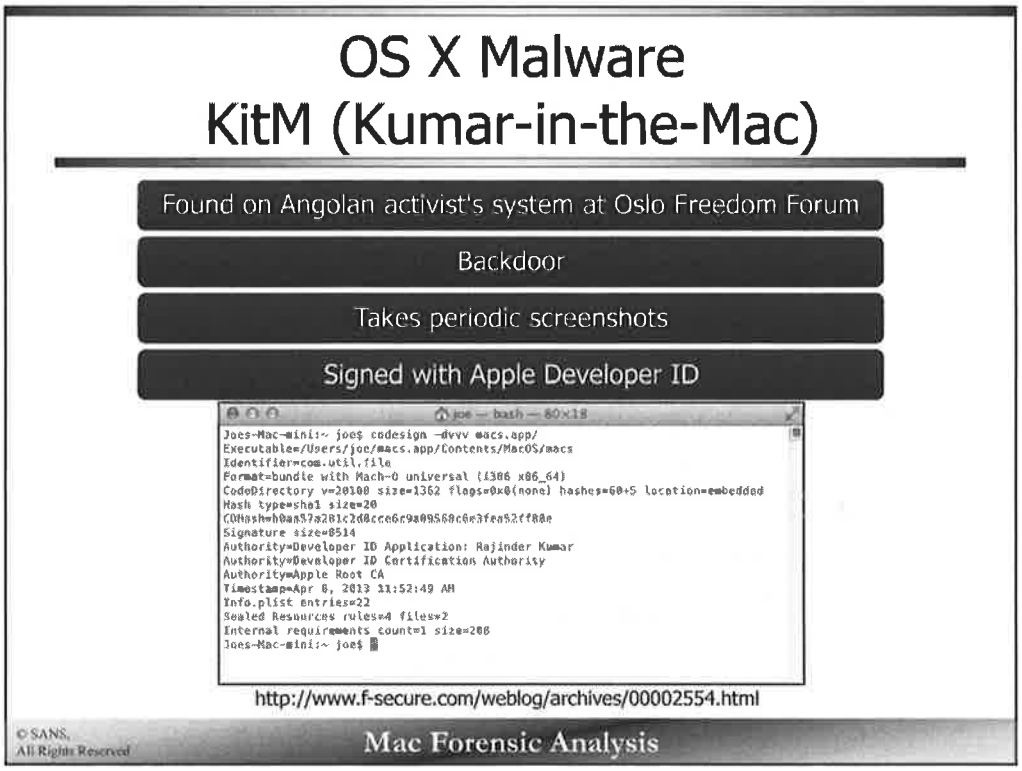
http://nakedsecurity.sophos.com/2012/07/25/mac-malware-crisis-on-mountain-lion-eve/

http://nakedsecurity.sophos.com/2012/07/26/mac-malware-spies-morcut-crisis/

http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/OSX~Morcut-A/detailed-analysis.aspx

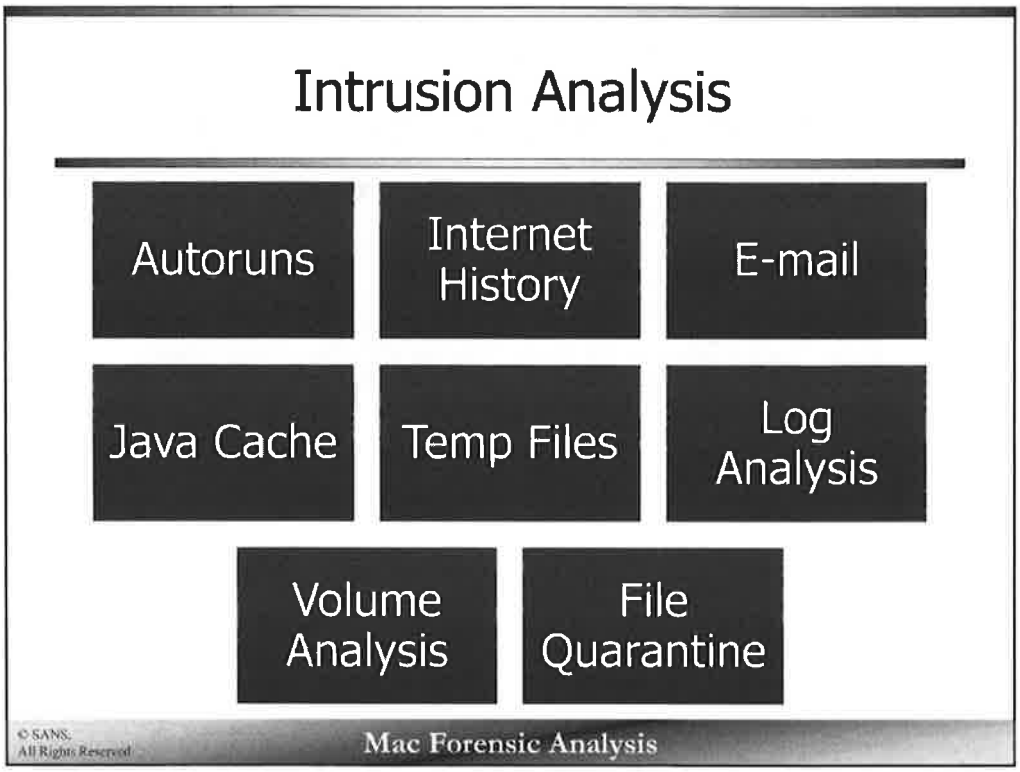
http://www.intego.com/mac-security-blog/new-apple-mac-trojan-called-osxcrisis-discovered-by-intego-virus-team/

http://www.symantec.com/connect/blogs/crisis-windows-sneaks-virtual-machines



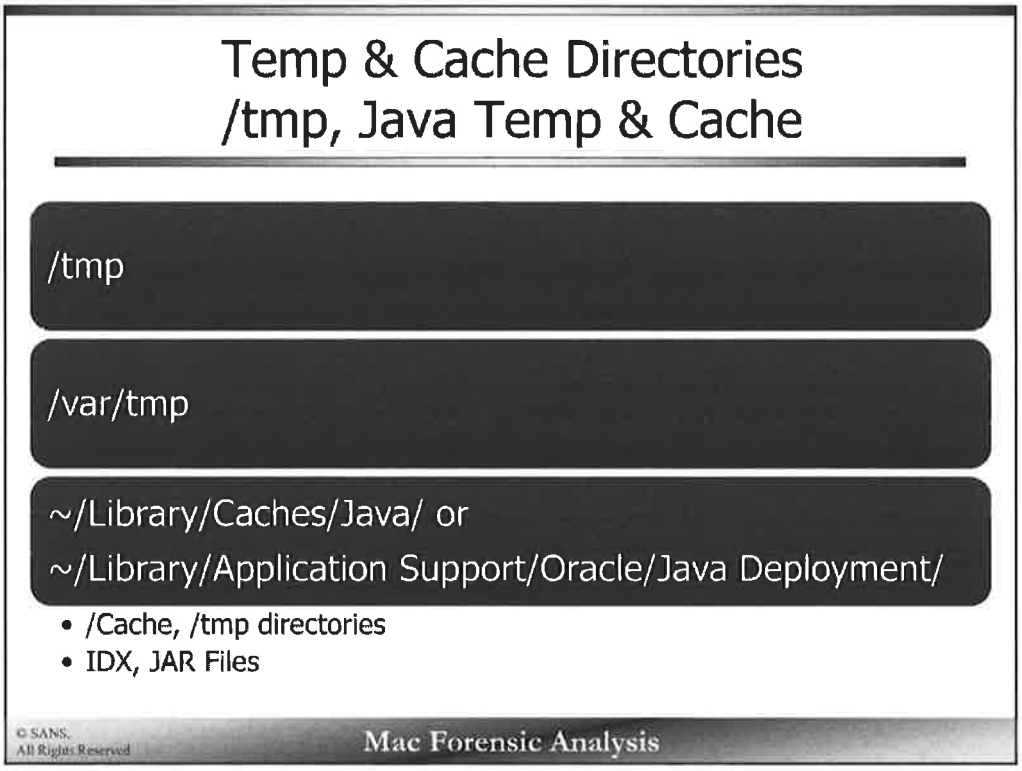
The Kumar-In-The-Mac or KitM malware was unique because it was the first major piece of malware that was found signed by a legitimate Apple Developer ID.

This malware was found on another activist system at the Oslo Freedom Forum in Norway. The malware has simple backdoor access and has the ability to take periodic screenshots of the system.



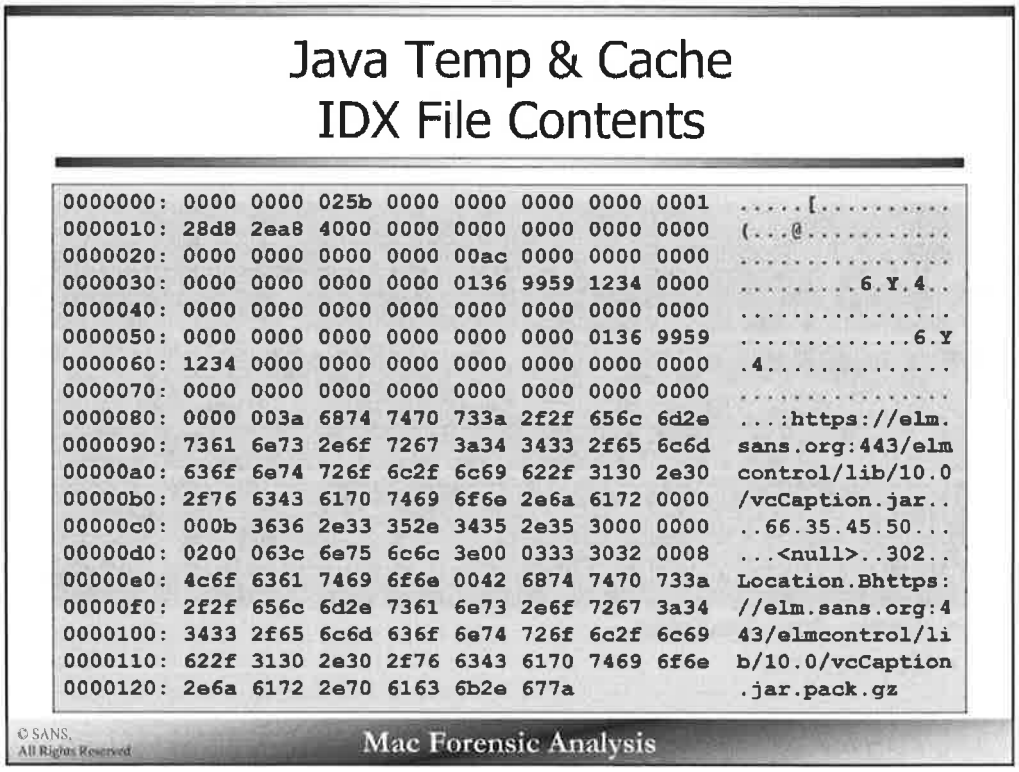
SANS FOR508 does a really great job of walking an investigator through a Windows intrusion analysis. We can use these same processes and techniques on a Mac intrusion analysis. The only difference is the file types and tools used.

While most of these topics have already been covered in the class, such as Mac autoruns, Internet and e-mail, log analysis, and volume analysis – there are a few more areas specific to finding Mac malware on a system. These topics include temporary directories, Java cache files, and the file quarantine process.

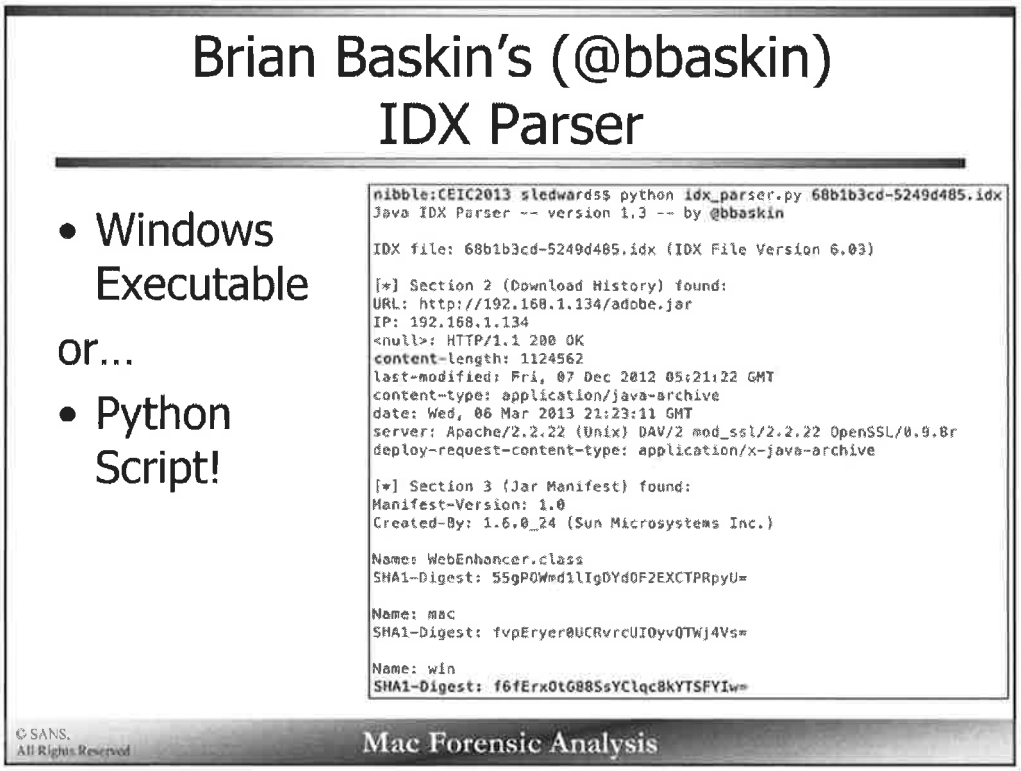


Temporary directories such as `/tmp` and `/var/tmp` have a tendency to be locations where malware files or decoy documents are written when malware is executed on the system, likely because they have permissions to write in these locations.

Java cache files located in the `~/Library/Caches/Java/` directory may contain Java IDX and JAR files. These files may need further investigations to determine if a Java-based piece of malware was executed on the system. Java ARchive files, (JAR) files, are the binary that is executed on the system. The IDX files are a Java index file that shows where the JAR file came from and other metadata.



Java IDX files contain data such as IP or web address showing where the Java JAR file was downloaded from, when it was downloaded, how large it is, file checksums, and the contents of the Java archive. Only some of the data can be parsed by the human eye in the hex view of the file.



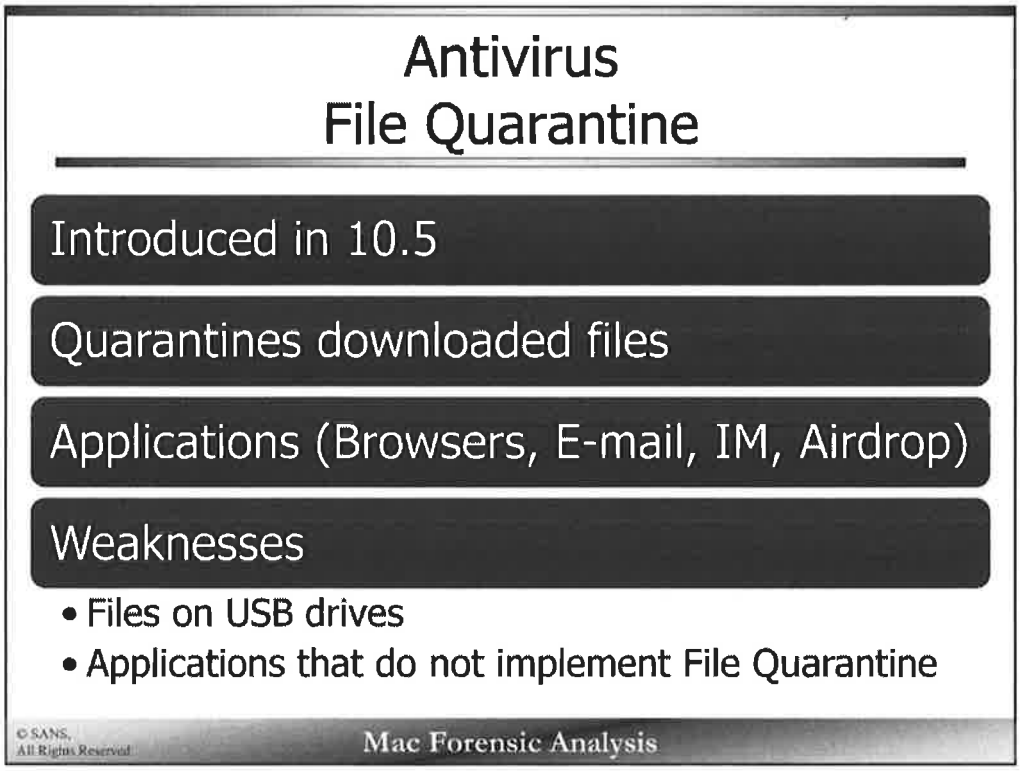
Brian Baskin (`@bbaskin`) developed a great IDX file parser that can be used on Windows platforms with a windows executable or on most other platforms with a Python script.

The screenshot shows an example of an IDX file that contains information for the JAR file related to the Crisis malware. The JAR file, named `adobe.jar` was downloaded from `http://192.168.1.134` on March 6th, 2013. This JAR file contains the `WebEnhancer.class` file as well as cross-platform binaries named '`mac`' and '`win`'.

References:

http://github.com/Rurik/Java_IDX_Parser

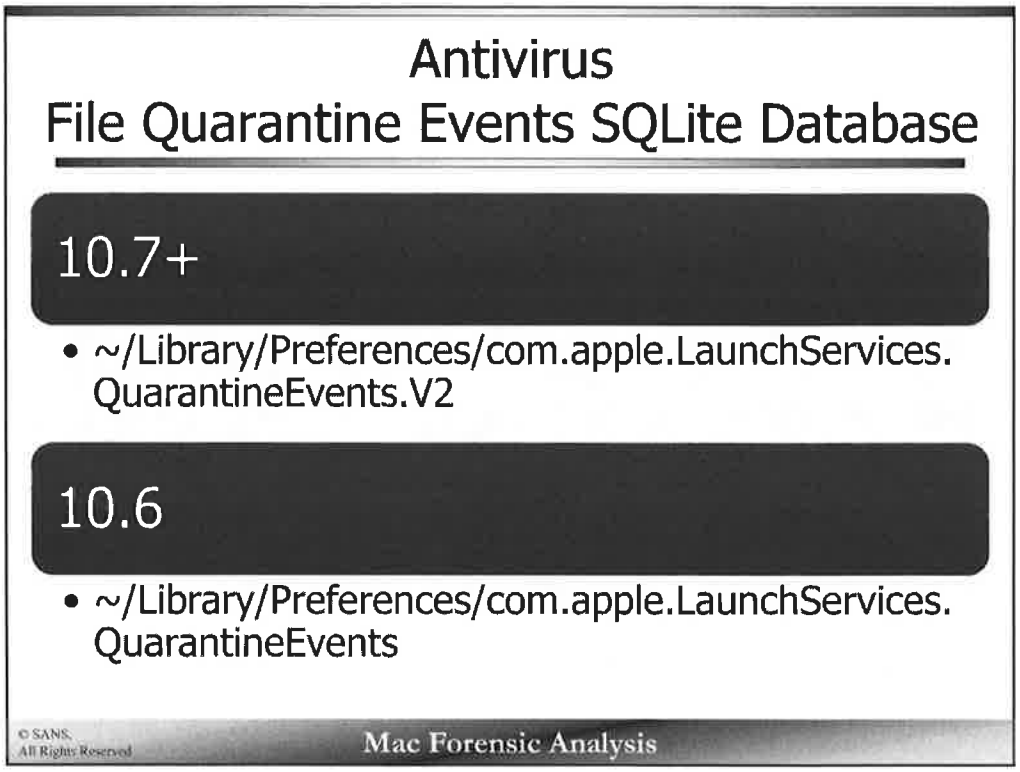
```
nibble:CEIC2013 sledwards$ python idx_parser.py 68b1b3cd-5249d485.idx
Java IDX Parser -- version 1.3 -- by @bbaskin

IDX file: 68b1b3cd-5249d485.idx (IDX File Version 6.03)

[*] Section 2 (Download History) found:
URL: http://192.168.1.134/adobe.jar
IP: 192.168.1.134
<null>: HTTP/1.1 200 OK
content-length: 1124562
last-modified: Fri, 07 Dec 2012 05:21:22 GMT
content-type: application/java-archive
date: Wed, 06 Mar 2013 21:23:11 GMT
server: Apache/2.2.22 (Unix) DAV/2 mod_ssl/2.2.22 OpenSSL/0.9.8r
deploy-request-content-type: application/x-java-archive

[*] Section 3 (Jar Manifest) found:
Manifest-Version: 1.0
Created-By: 1.6.0_24 (Sun Microsystems Inc.)

Name: WebEnhancer.class
SHA1-Digest: 55gPOWmd1lIgDYdOF2EXCTPRpyU=

Name: mac
SHA1-Digest: fvpEryer0UCRvrcUIOyvQTWj4Vs=

Name: win
SHA1-Digest: f6fErx0tGB8SsYClqc8kYTSFYIw=
```



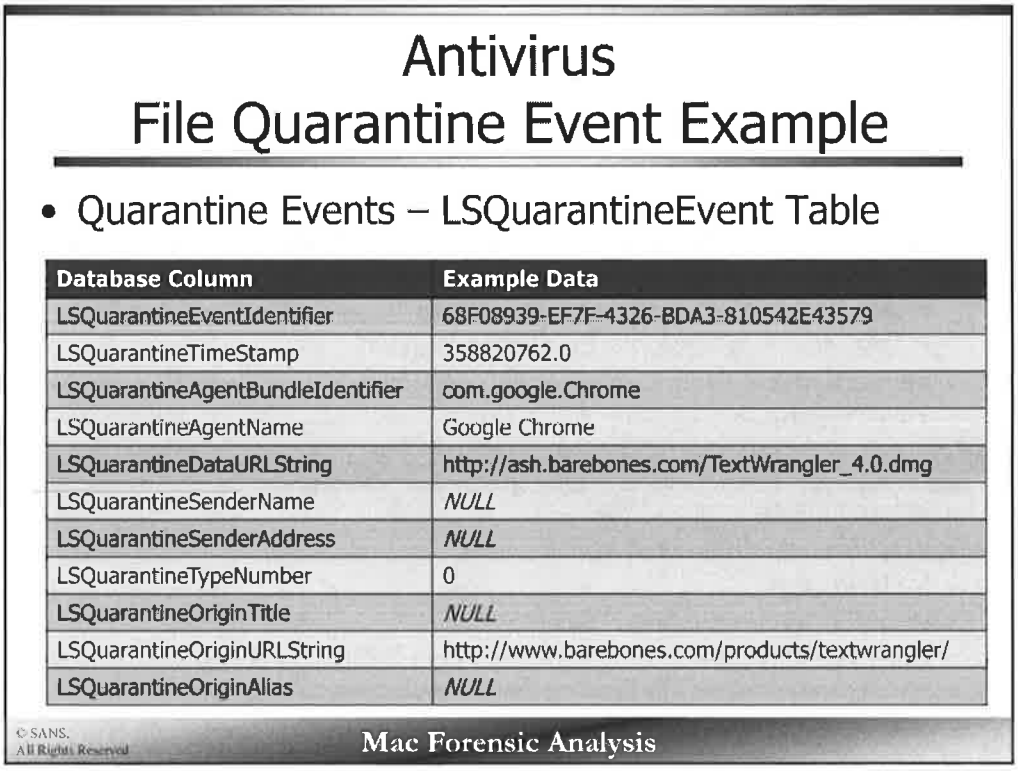
Apple introduced the concept of file quarantining in 10.5. It is a method that OS X uses to tag a file with where it came from so it can be checked by the Xprotect anti-virus solution.

Files get quarantined if they are downloaded by applications that implement this feature. This can be checked by looking at the `LSFileQuarantineEnabled` key in an Applications `Info.plist` file in the Application bundle. If it is implemented, it will be set to '`True`'. Most popular web browsers and e-mail clients implement this on the Mac today.

The File Quarantine method does have weaknesses. For instance, it does not quarantine files that are copied off of thumb drives, or from applications that do not implement the File Quarantine functionality.



A SQLite database containing file quarantine data is located in the user's `~/Library/Preferences` directory. On 10.6 systems, the database is named `com.apple.LaunchServices.QuarantineEvents`, while on systems 10.7 and above, it has been upgraded to `com.apple.LaunchServices.QuarantineEvents.V2`.

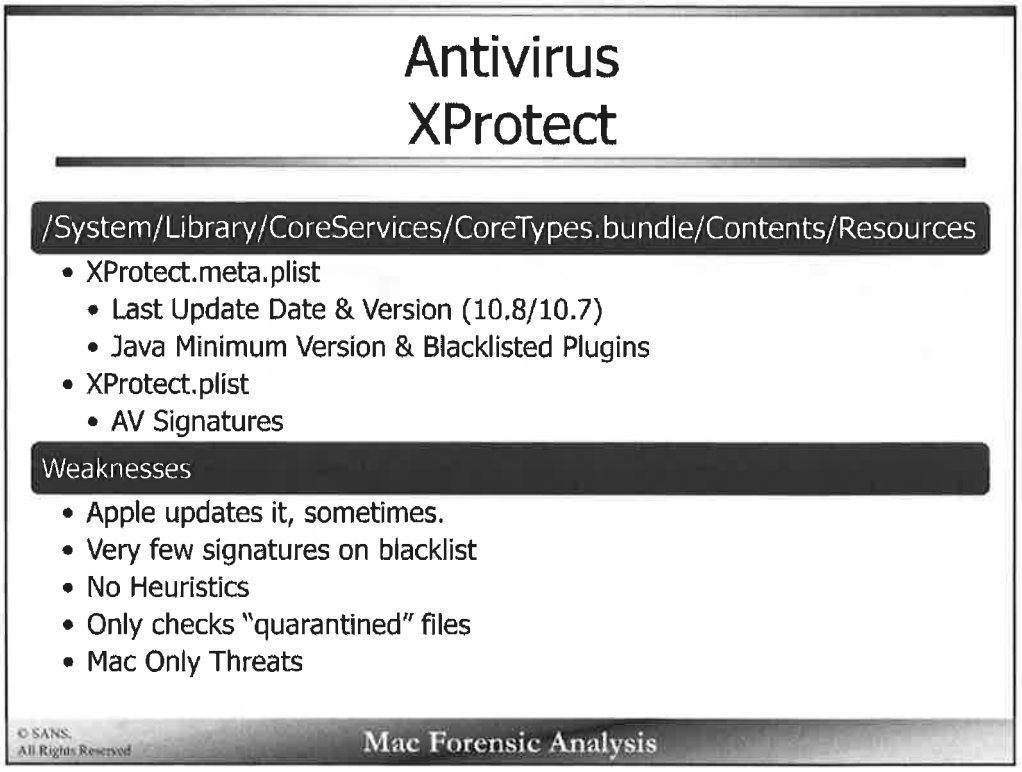


An example of a record in the File Quarantine Events database is shown above. This record has been extracted from the database and placed into this table for easier reading.

The information in the database can help an analyst determine where a certain file was downloaded from with related information.

- `LSQuarantineEventIdentifier` Unique Event Identifier GUID
- `LSQuarantineTimeStamp` – Timestamp when file was quarantined (Mac Absolute Time/WebKit time, seconds from 1/1/2001)
- `LSQuarantineAgentBundleIdentifier` – Bundle ID of the application that downloaded the file
- `LSQuarantineAgentName` – Application that downloaded the file (e.g., Safari, Google Chrome, Mail)
- `LSQuarantineDataURLString` – URL the file was actually downloaded from (may be different from the URL used by the user to download the file)
- `LSQuarantineSenderName` – Files downloaded from the Mail applications include the e-mail sender's name. Those downloaded from Airdrop will have the hostname sender system
- `LSQuarantineSenderAddress` - Used in files downloaded from Mail application, E-mail Sender's E-mail Address
- `LSQuarantineTypeNumber` – Quarantine Type
  - 0 – Web Browsers
  - 1 – Xcode
  - 2 – Apple Mail
  - 6 - Airdrop
- `LSQuarantineOriginTitle` – Used in files downloaded from Mail, E-mail Subject
- `LSQuarantineOriginURLString` – URL the user visited to download the file (Browser); E-mail Server Information
- `LSQuarantineOriginAlias` – Unknown

Files sent via Airdrop will have the Agent Name "NetworkBrowserAgent

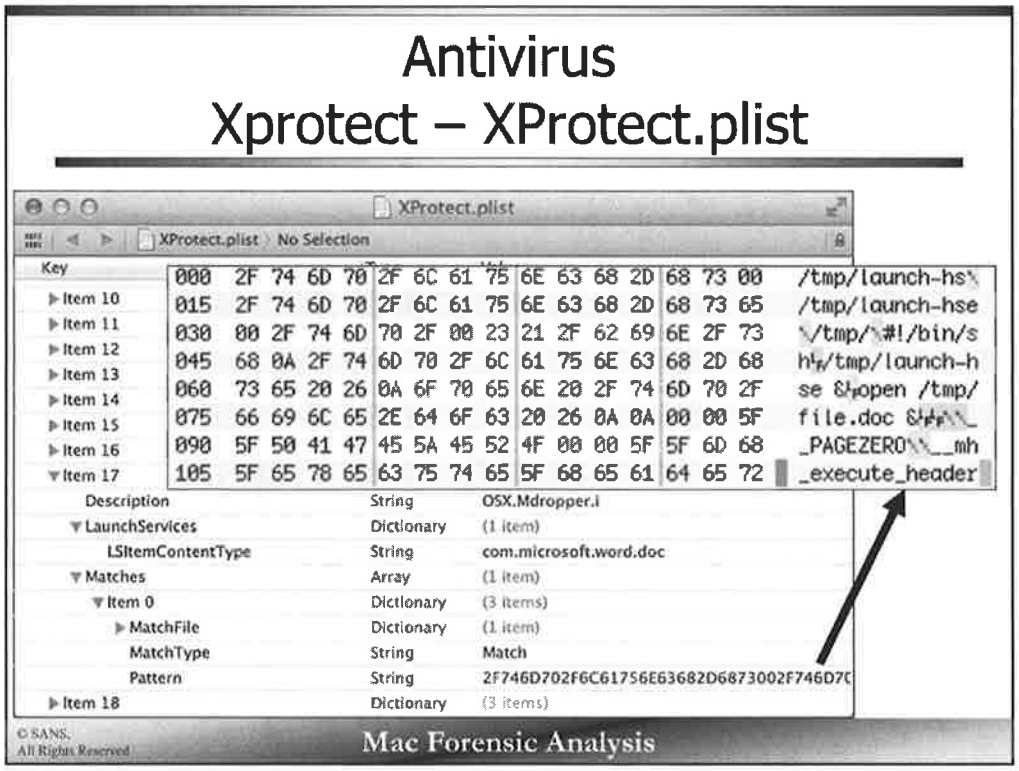


XProtect was introduced in 10.6. The XProtect system is Apple's answer to anti-virus. Xprotect uses the `Xprotect.plist` property list located in the `/System/Library/CoreServices/CoreTypes.bundle/Contents/Resources/` directory. This property list contains signatures of well known threats for the Mac. The `XProtect.meta.plist` file located in the same directory contains the date when this signature property list was last updated.

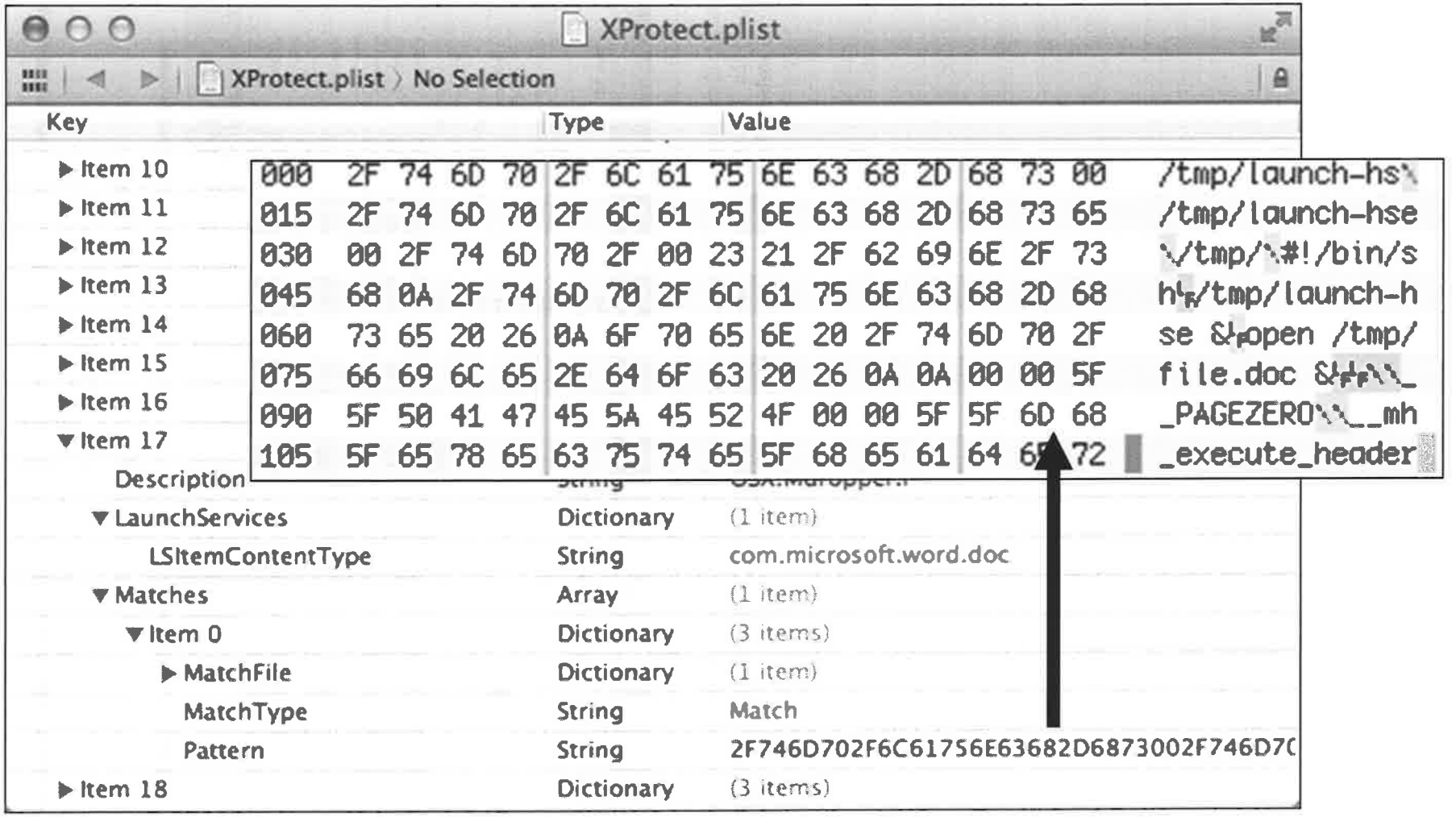
The XProtect system does have its weaknesses. For instance, it only gets updated when Apple decides to update it. It is not meant to be updated by the user.

The signatures are limited, not only in quantity but in quality. These signatures do not use heuristics and only serve to protect from Mac-related threats. Most Windows anti-virus products will scan for threats to other operating systems.

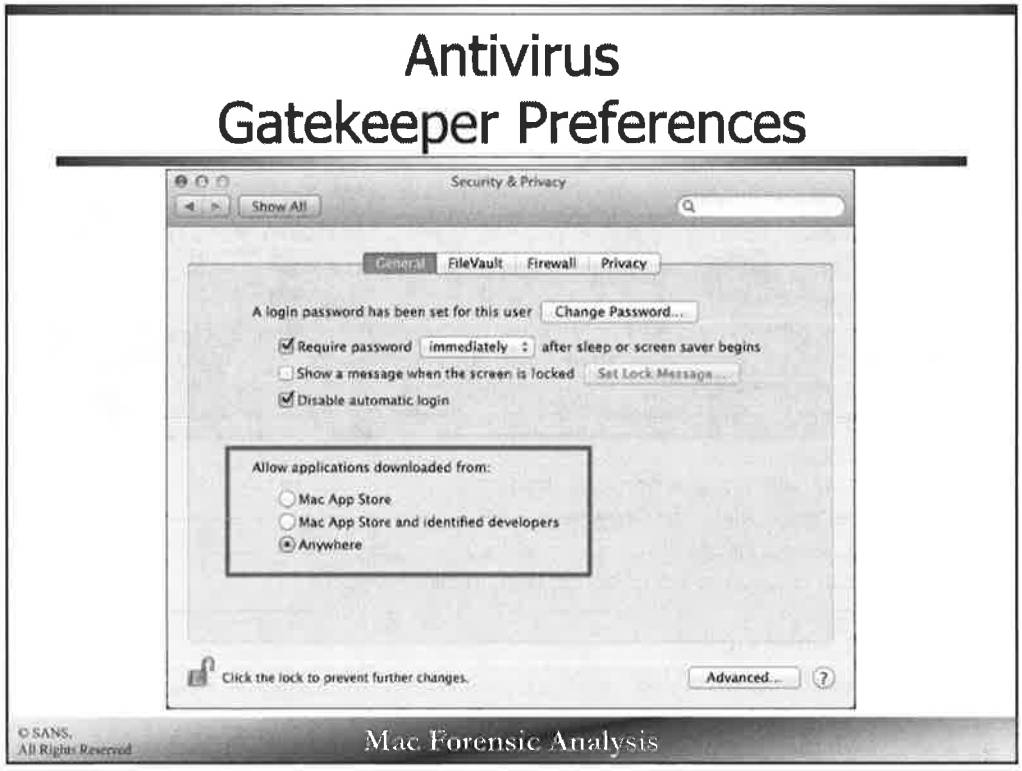
The Xprotect system only checks those files that are file quarantined. Other files transferred from USB drives or from network shares will not be checked.



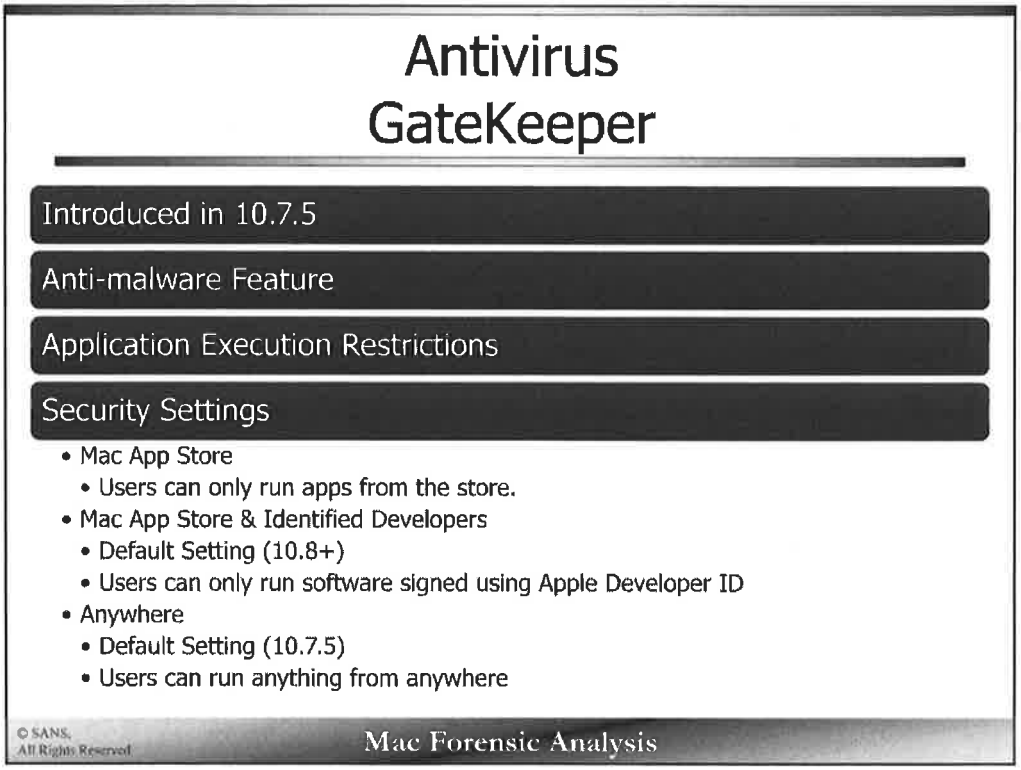
The screenshots above show the `XProtect.plist` property list file. The background screenshot shows the whole property list. The inset screenshot shows an example of the of one of the signatures found for the `MacControl` malware located in the `Pattern` key.


XProtect.plist
XProtect.plist 〉 No Selection
Key
Type
Value
Item 10
Item 11
Item 12
Item 13
Item 14
Item 15
Item 16
Item 17
Description
LaunchServices
LSItemContentType
Matches
Item 0
MatchFile
MatchType
Pattern
Item 18
Dictionary
(1 item)
String
com.microsoft.word.doc
Array
(1 item)
Dictionary
(3 items)
Dictionary
(1 item)
String
Match
String
2F746D702F6C61756E63682D6873002F746D7C
Dictionary
(3 items)
000 2F 74 6D 70 2F 6C 61 75 6E 63 68 2D 68 73 00 /tmp/launch-hs
015 2F 74 6D 70 2F 6C 61 75 6E 63 68 2D 68 73 65 /tmp/launch-hse
030 00 2F 74 6D 70 2F 00 23 21 2F 62 69 6E 2F 73 /tmp/#!/bin/s
045 68 0A 2F 74 6D 70 2F 6C 61 75 6E 63 68 2D 68 h/tmp/launch-h
060 73 65 20 26 0A 6F 70 65 6E 20 2F 74 6D 70 2F se &open /tmp/
075 66 69 6C 65 2E 64 6F 63 20 26 0A 0A 00 00 5F file.doc &__
090 5F 50 41 47 45 5A 45 52 4F 00 00 5F 5F 6D 68 _PAGEZERO__mh
105 5F 65 78 65 63 75 74 65 5F 68 65 61 64 65 72 _execute_header

138

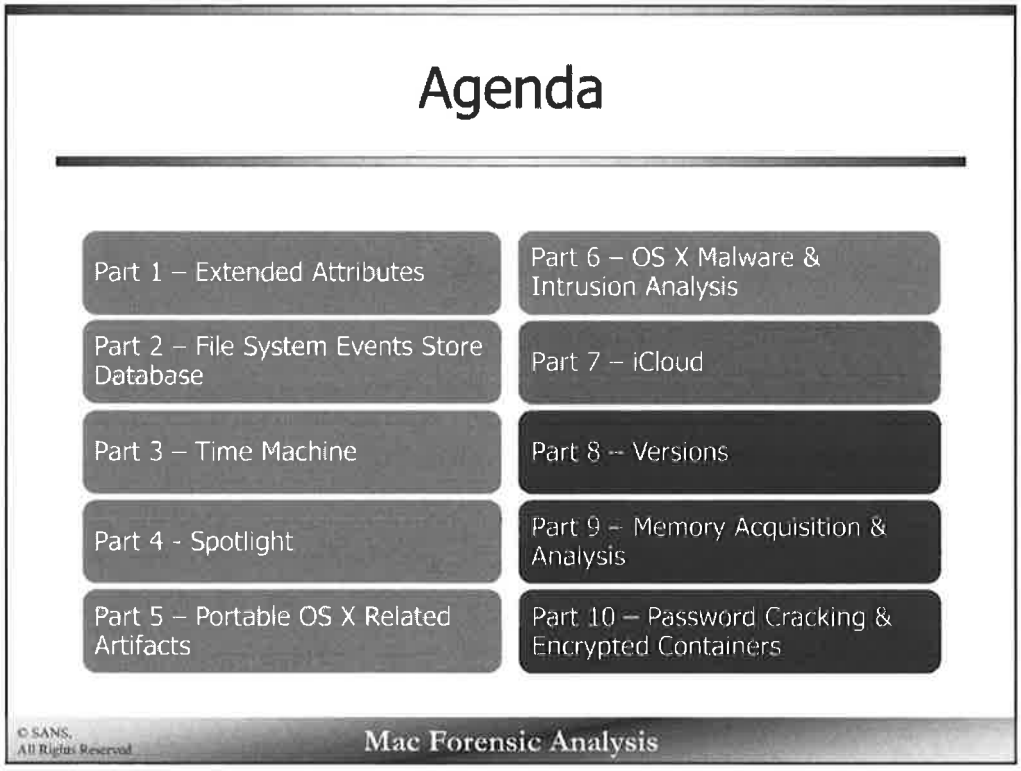


The screenshot shows the Gatekeeper settings in the Security & Privacy Preferences panel.

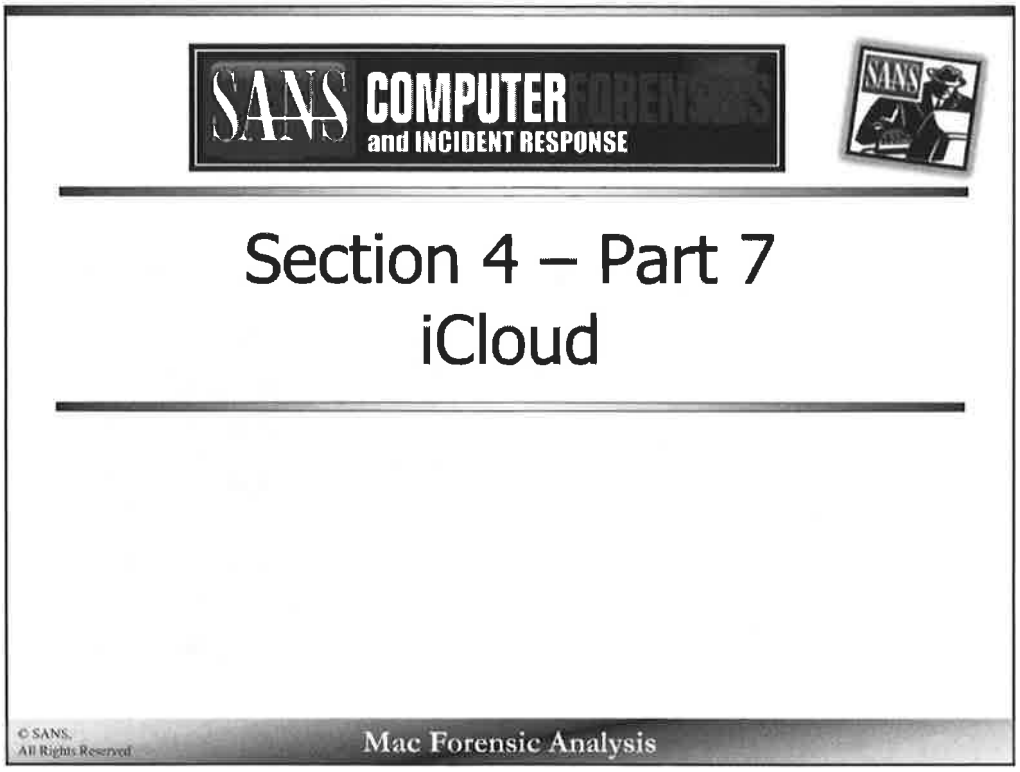


Gatekeeper was introduced 10.7.5. This functionality allowed the user to choose the level of security used when downloading applications from the Internet. Three options existed for the user to choose from.

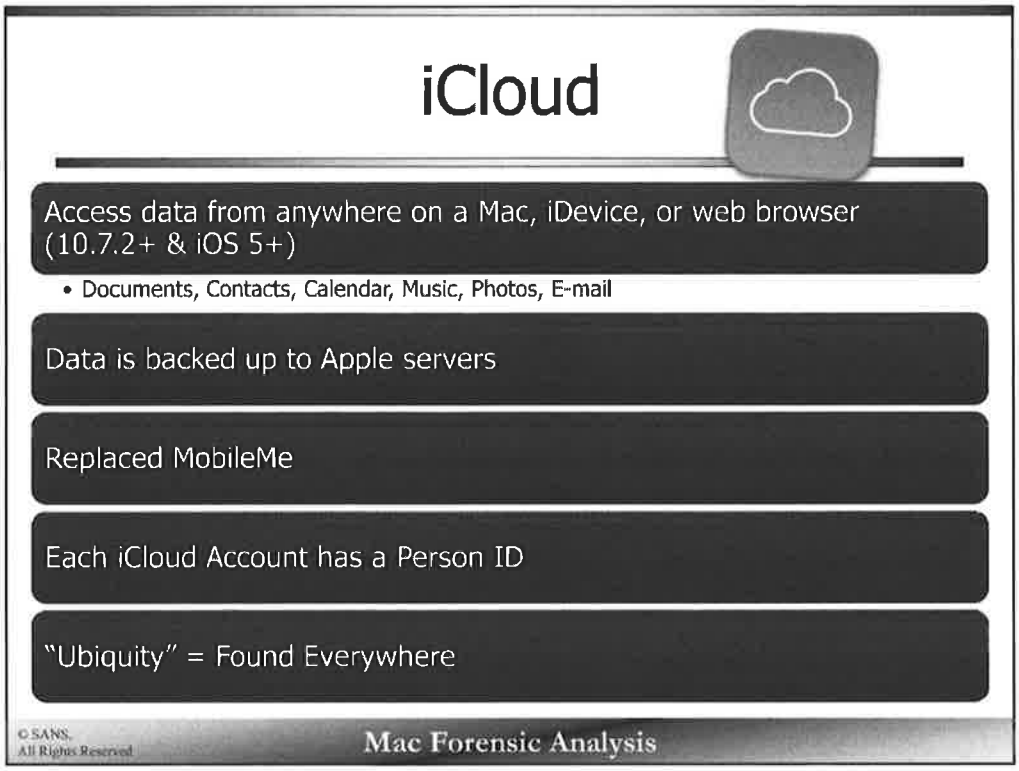
- Mac App Store – Only allow applications to open that were downloaded from the Mac App Store
- Mac App Store and identified developers – Default selection in 10.8+. This limits applications to open only if they came from the Mac App Store or signed with an Apple Developer ID
- Anywhere – Default selection in 10.7.5. This will allow applications downloaded from anywhere to open.



This page intentionally left blank.



This page intentionally left blank.



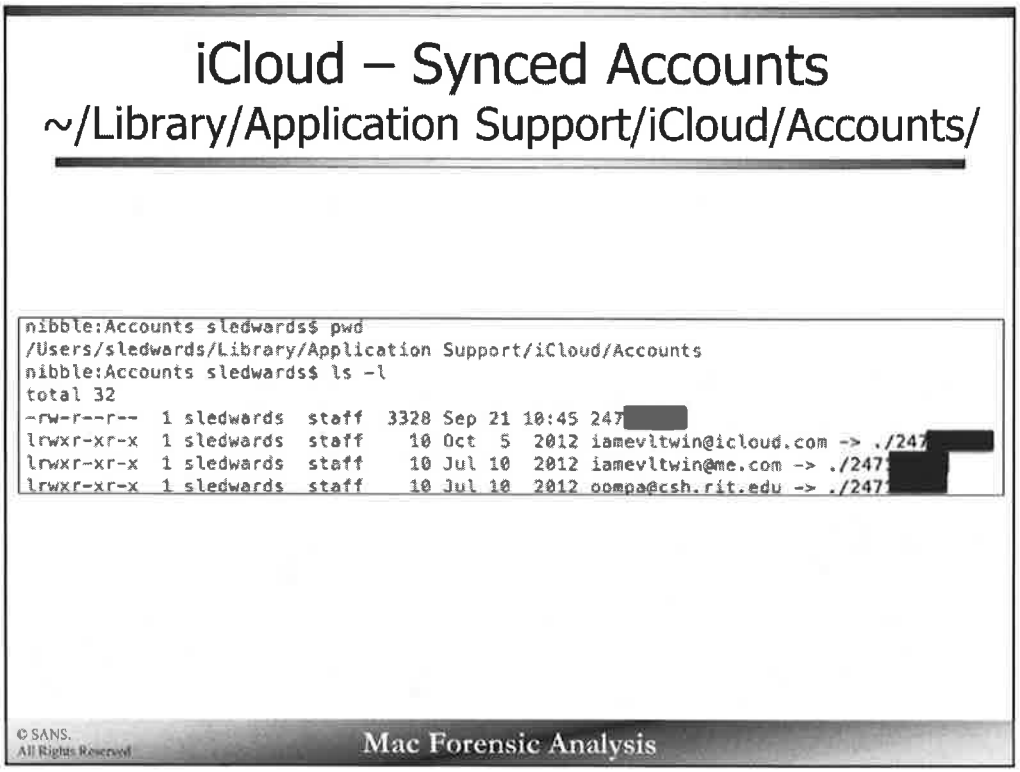
iCloud is the term used for Apple's cloud services. iCloud has the ability to access data anywhere including documents, contacts, calendar, music, photos, and e-mail.

iCloud can be accessed natively from any OS X system (10.7.2+) , iOS device (5+), or from a web browser (via iCloud.com). This data is stored on Apple servers. Rumor is that they may be using Amazon and Microsoft cloud servers on the backend.

iCloud replaces the previous incarnation of Apple cloud services MobileMe when it came out in 10.7.2.

Each iCloud account uses a unique identifier known as a Person ID.

One term you will see often in almost everything related to iCloud is "Ubiquity" which means "found everywhere". iCloud documents, e-mail, contacts, etc., are meant to be accessible everywhere.

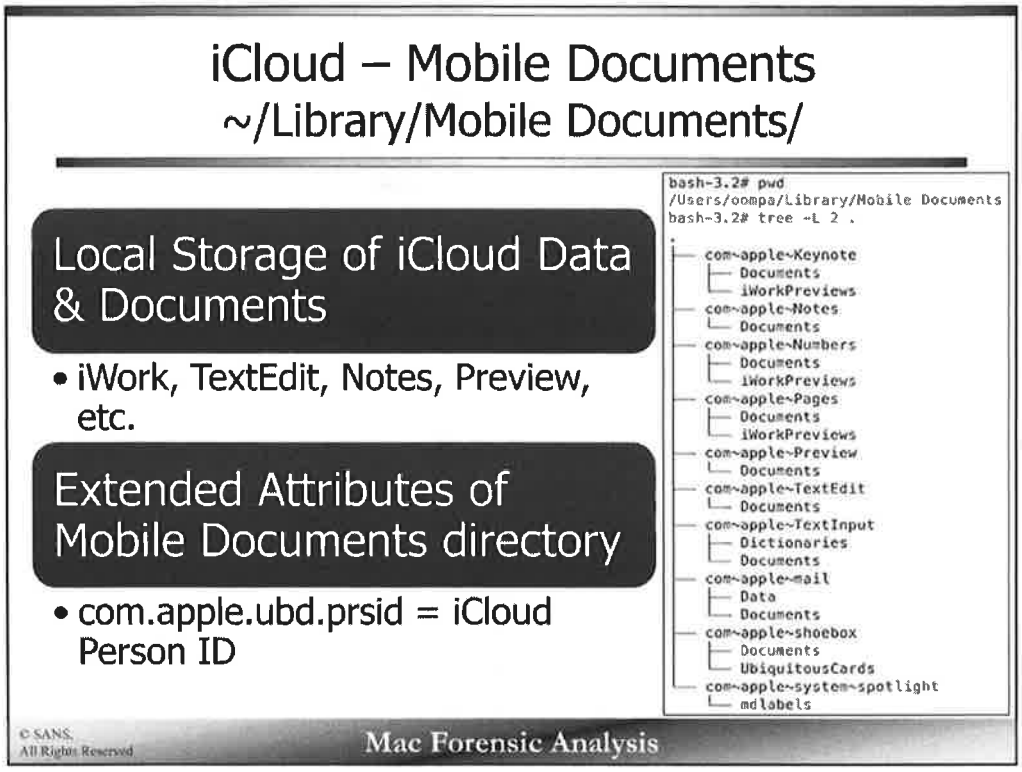


The iCloud Person ID number is the unique number associated with a user's iCloud account.

Shown in the screenshot above are the files located in the users `~/Library/Application Support/iCloud/Accounts/` directory. This directory contains a file with a numeric filename. This number is the user's iCloud Person ID. This iCloud ID has been redacted for privacy reasons.

This directory also contains link files that point to the numeric filename. Each link is an e-mail associated with that user's iCloud account. This iCloud account has been associated with three e-mail accounts.

```
nibble:Accounts sledwards$ pwd
/Users/sledwards/Library/Application Support/iCloud/Accounts
nibble:Accounts sledwards$ ls -l
total 32
-rw-r--r--  1 sledwards  staff  3328 Sep 21 10:45 247[redacted]
lrwxr-xr-x  1 sledwards  staff    10 Oct  5  2012 iamevltwin@icloud.com -> ./247[redacted]
lrwxr-xr-x  1 sledwards  staff    10 Jul 10  2012 iamevltwin@me.com -> ./247[redacted]
lrwxr-xr-x  1 sledwards  staff    10 Jul 10  2012 oompa@csh.rit.edu -> ./247[redacted]
```

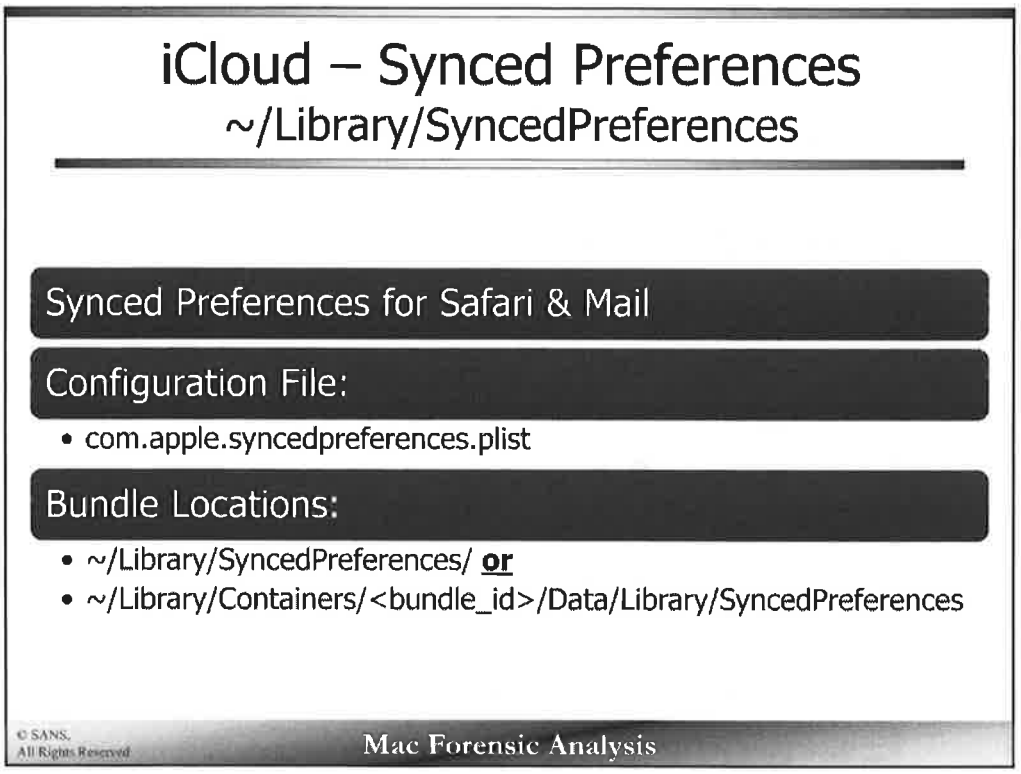


iCloud has the ability to store documents from a variety of applications including iWork, TextEdit, and Notes. Copies of these files are stored on an OS X system in the `~/Library/Mobile Documents/` directory.

The screenshot shows a tree command output of the `Mobile Documents` directory. Each application has its own directory named with the reverse DNS naming scheme as shown before, but with tildes (~) instead of periods. For example the TextEdit documents are stored in a directory named `com~apple~TextEdit`. The documents will be stored in the `Documents` sub-directory. iWork applications (Pages, Numbers, Keynote) contain another sub-directory, `iWorkPreviews`, that contain JPG document previews of each document.

Another way to determine an iCloud Person ID from these documents is to view their extended attributes. The attribute `com.apple.ubd.prsid` contains this numeric identifier.

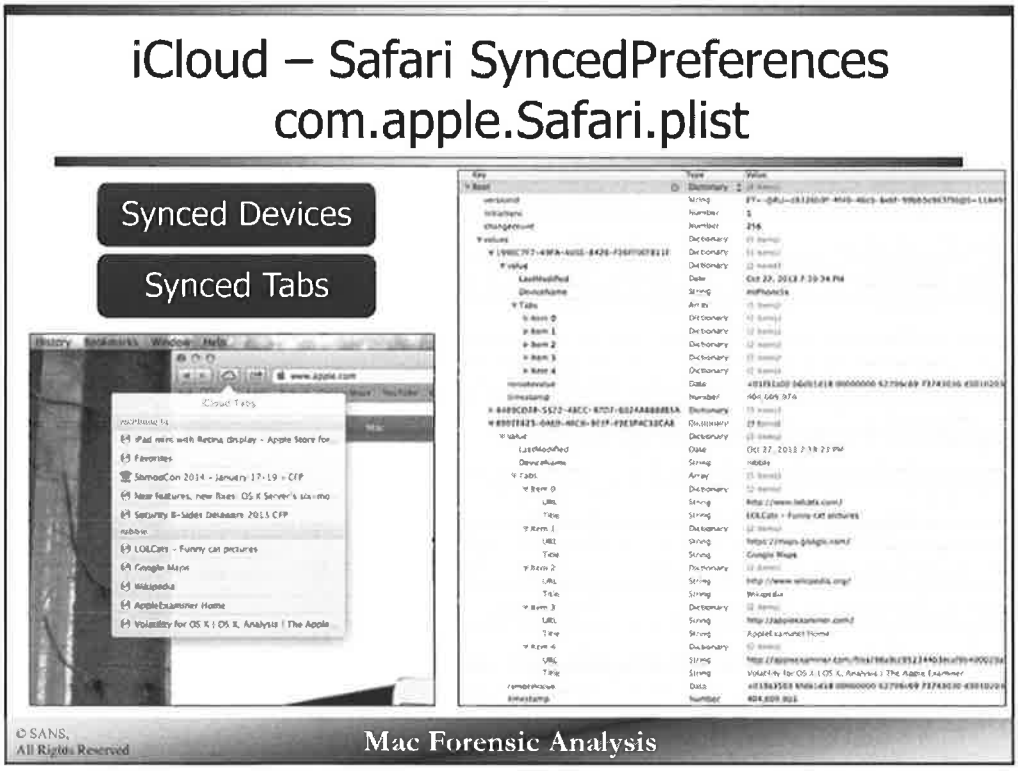
```
bash-3.2# pwd
/Users/oompa/Library/Mobile Documents
bash-3.2# tree -L 2 .
.
├── com~apple~Keynote
│   ├── Documents
│   └── iWorkPreviews
├── com~apple~Notes
│   └── Documents
├── com~apple~Numbers
│   ├── Documents
│   └── iWorkPreviews
├── com~apple~Pages
│   ├── Documents
│   └── iWorkPreviews
├── com~apple~Preview
│   └── Documents
├── com~apple~TextEdit
│   └── Documents
├── com~apple~TextInput
│   ├── Dictionaries
│   └── Documents
├── com~apple~mail
│   ├── Data
│   └── Documents
├── com~apple~shoebox
│   ├── Documents
│   └── UbiquitousCards
└── com~apple~system~spotlight
    └── mdlabels
```



iCloud sync preferences are stored in the `~/Library/SyncedPreferences/` directory. These are the technical details on how iCloud syncs with the Apple servers to sync the data across all devices.

The main configuration data can be found in the `com.apple.syncedpreferences.plist` file.

The Safari and Mail applications each have their own property list with their associated preferences. These files may be located in either the `~/Library/SyncedPreferences/` directory itself or in the sandbox container directory located in the `~/Library/Containers/<bundle_id>/Data/Library/SyncedPreferences/`.



The synced preferences for the Safari application are found in the `com.apple.Safari.plist` file. While there are many keys, those of particular interest to a forensic investigator are those that are associated with the synced devices and Safari iCloud tabs.

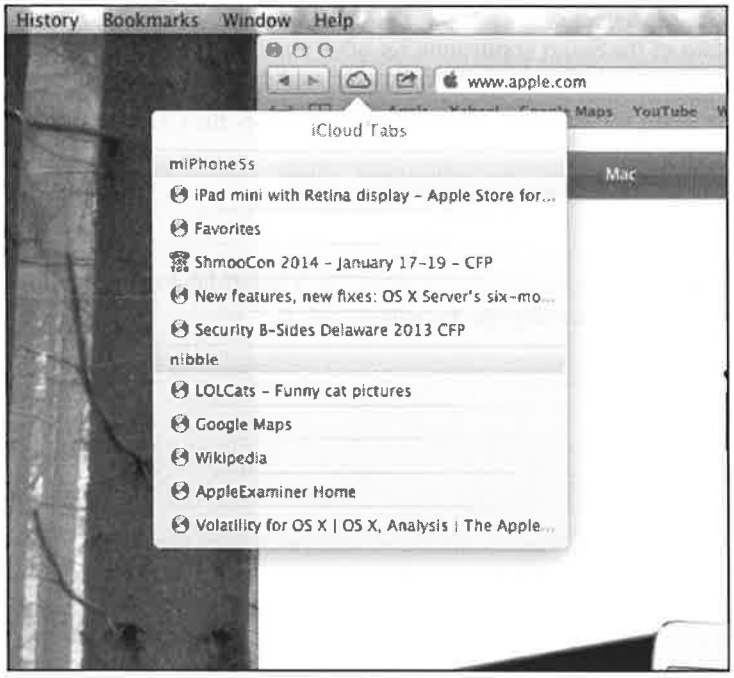
The screenshot on the left shows what these iCloud Tabs look like in a Safari web browser. Two other devices `miPhone5s` and `nibble` (an iPhone and laptop) are synced to this device (another computer).

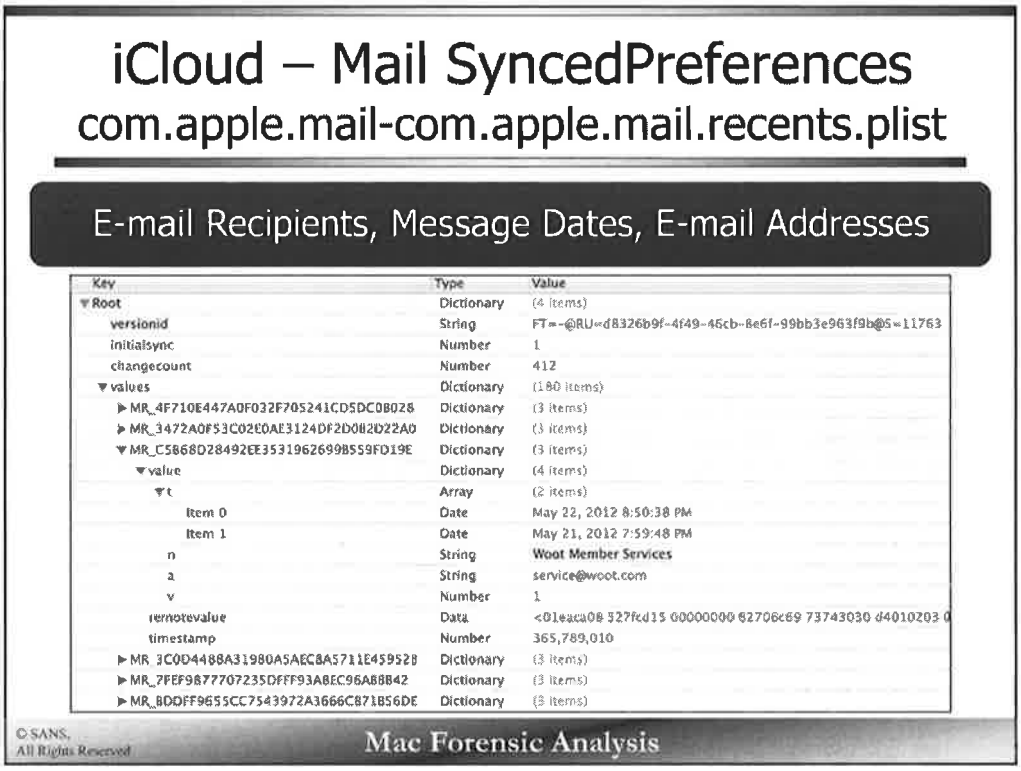
Safari stores the open tabs of the Safari application for other computers or iDevices. In this file, under a GUID for each synced device, is a `value` key with the following subkeys:

- `LastModified` – Last modification of the data inside the GUID key (same as the `timestamp` subkey)
- `DeviceName` – Synced device name (i.e., hostname)

Also stored here are each tab, under the `Tabs` key array. Each Safari tab has a stored `URL` and `Title` key that contains the URL and the title of the webpage associated with it.

Key	Type	Value
▼ Root	Dictionary	(4 items)
versionid	String	FT=-@RU=d8326b9f-4f49-46cb-8e6f-99bb3e963f9b@S=11849
initialsync	Number	1
changecount	Number	256
▼ values	Dictionary	(3 items)
▼ 199EC7F7-49FA-405E-8426-F26FF0EF811F	Dictionary	(3 items)
▼ value	Dictionary	(3 items)
LastModified	Date	Oct 27, 2013 7:39:34 PM
DeviceName	String	miPhone5s
▼ Tabs	Array	(5 items)
▶ Item 0	Dictionary	(2 items)
▶ Item 1	Dictionary	(2 items)
▶ Item 2	Dictionary	(2 items)
▶ Item 3	Dictionary	(2 items)
▶ Item 4	Dictionary	(2 items)
remotevalue	Data	<01f91a00 b6db1d18 00000000 62706c69 73743030 d3010203
timestamp	Number	404,609,974
▶ 8489CD2B-5522-48CC-87D7-6024A8888B5A	Dictionary	(3 items)
▼ 8907F825-0AE0-46C6-922F-F3E3F4C32CA8	Dictionary	(3 items)
▼ value	Dictionary	(3 items)
LastModified	Date	Oct 27, 2013 7:38:23 PM
DeviceName	String	nibble
▼ Tabs	Array	(5 items)
▼ Item 0	Dictionary	(2 items)
URL	String	http://www.lolcats.com/
Title	String	LOLCats - Funny cat pictures
▼ Item 1	Dictionary	(2 items)
URL	String	https://maps.google.com/
Title	String	Google Maps
▼ Item 2	Dictionary	(2 items)
URL	String	http://www.wikipedia.org/
Title	String	Wikipedia
▼ Item 3	Dictionary	(2 items)
URL	String	http://appleexaminer.com/
Title	String	AppleExaminer Home
▼ Item 4	Dictionary	(2 items)
URL	String	http://appleexaminer.com/files/98a9cc952344b3ecaf9b400029a
Title	String	Volatility for OS X | OS X, Analysis | The Apple Examiner
remotevalue	Data	<019a3503 6fdb1d18 00000000 62706c69 73743030 d3010203
timestamp	Number	404,609,903



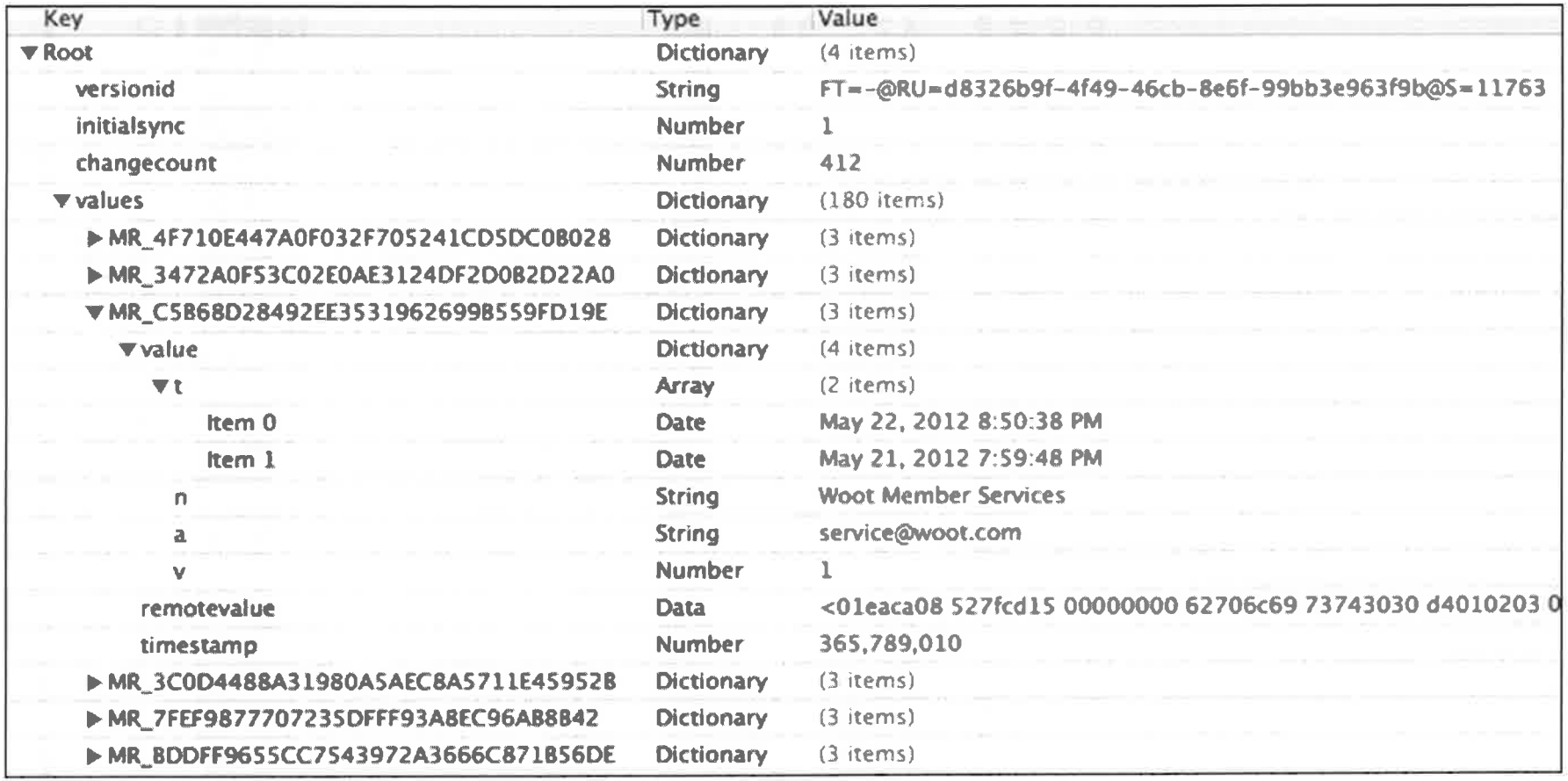


The synced preferences for the Mail application are found in a few property lists. The one shown above is the `com.apple.mail-com.apple.mail.recents.plist` property list file.

This property list contains data about the "recent" e-mail contacts. Fortunately for us, in reality, it contains even not-so-recent e-mail contacts! Note the value count for the `values` key – 180 items!

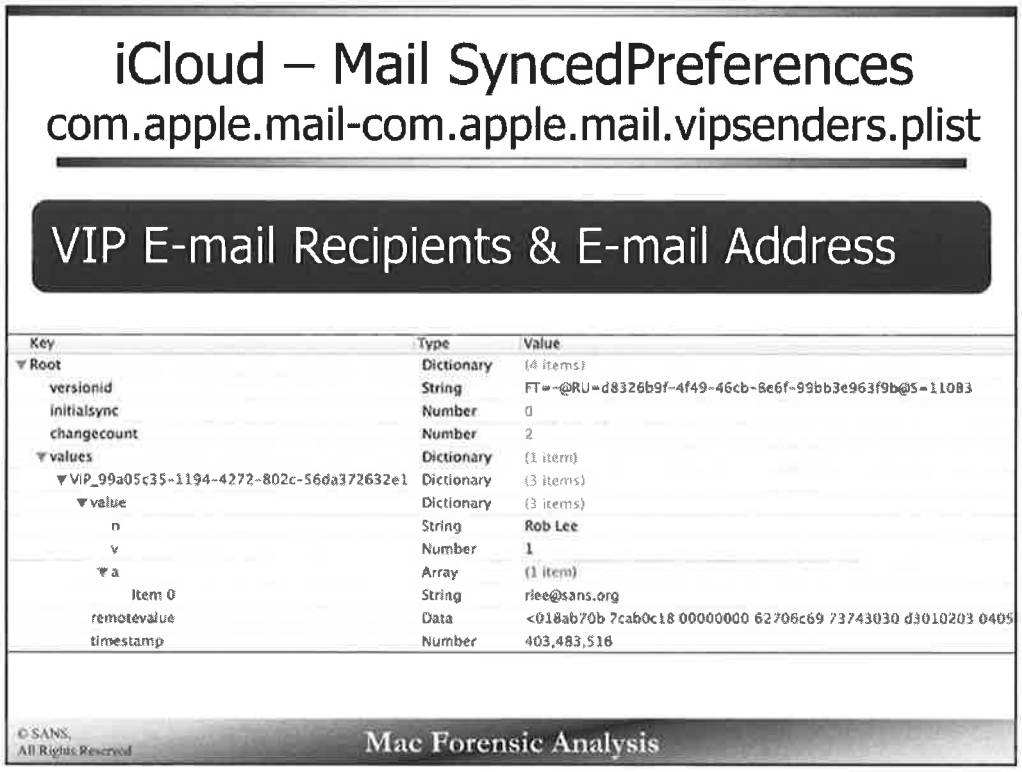
Each key under the values key (`MR_*`) contains information pertaining to an e-mail contact with multiple keys.

- `t` – Timestamps of previous e-mails
- `n` – Contact Name
- `a` – Contact's E-mail Address



Key	Type	Value
▼ Root	Dictionary	(4 items)
versionid	String	FT=-@RU=d8326b9f-4f49-46cb-8e6f-99bb3e963f9b@S=11763
initialsync	Number	1
changecount	Number	412
▼ values	Dictionary	(180 items)
▶ MR_4F710E447A0F032F705241CD5DC0B028	Dictionary	(3 items)
▶ MR_3472A0F53C02E0AE3124DF2D0B2D22A0	Dictionary	(3 items)
▼ MR_C5B68D28492EE3531962699B559FD19E	Dictionary	(3 items)
▼ value	Dictionary	(4 items)
▼ t	Array	(2 items)
Item 0	Date	May 22, 2012 8:50:38 PM
Item 1	Date	May 21, 2012 7:59:48 PM
n	String	Woot Member Services
a	String	service@woot.com
v	Number	1
remotevalue	Data	<01eaca08 527fcd15 00000000 62706c69 73743030 d4010203 0
timestamp	Number	365,789,010
▶ MR_3C0D4488A31980A5AEC8A5711E45952B	Dictionary	(3 items)
▶ MR_7FEF9877707235DFFF93A8EC96AB8B42	Dictionary	(3 items)
▶ MR_BDDFF9655CC7543972A3666C871B56DE	Dictionary	(3 items)

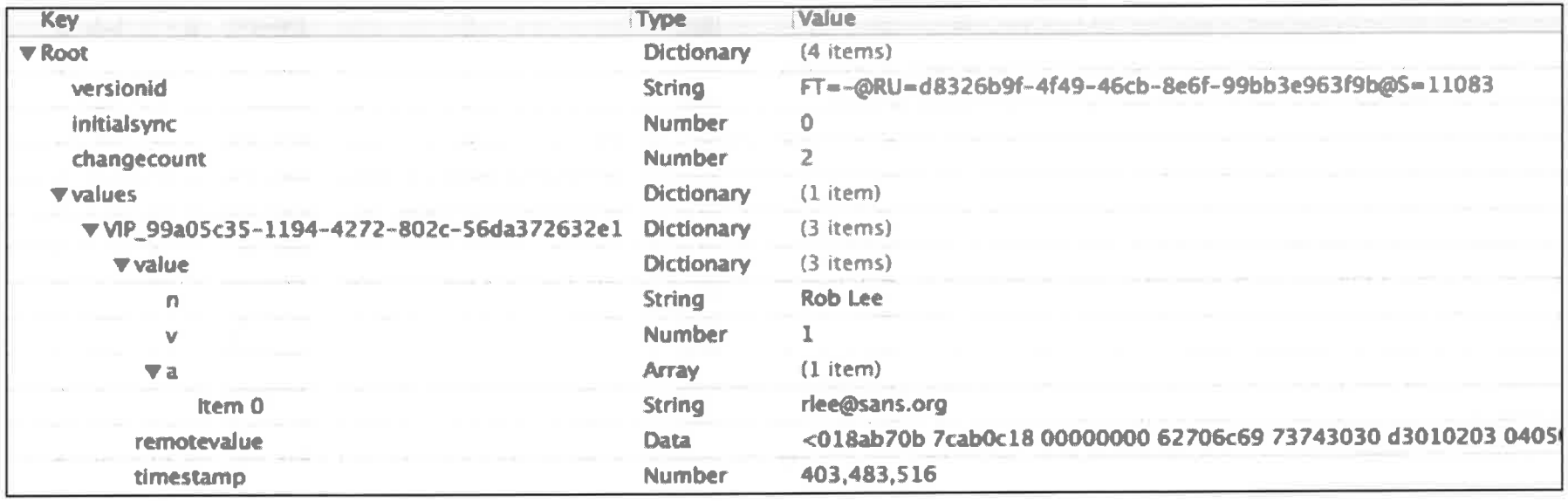
152



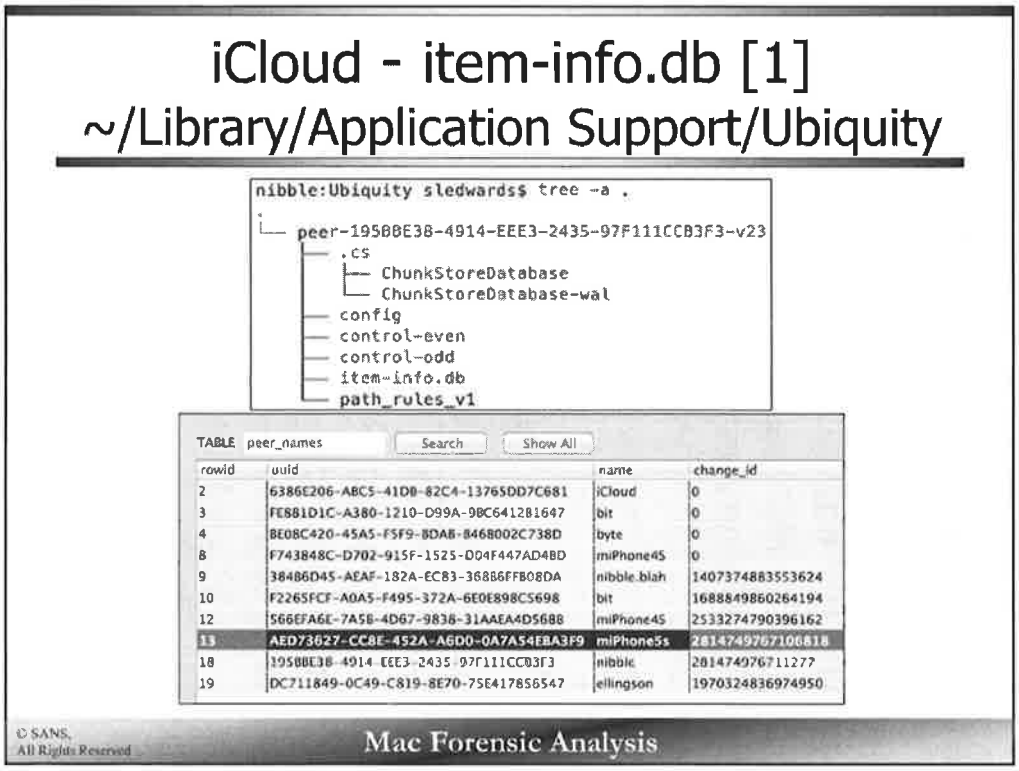
The `com.apple.mail com.apple.mail.vipsenders.plist` property list contains the contacts that the user marked as VIP e-mail contacts.

Under the key marked with a VIP (`VIP_*`) contains more single lettered keys:

- n – Contact Name
- a – Contact's E-mail Address



Key	Type	Value
▼Root	Dictionary	(4 items)
versionid	String	FT=-@RU=d8326b9f-4f49-46cb-8e6f-99bb3e963f9b@S=11083
initialsync	Number	0
changecount	Number	2
▼values	Dictionary	(1 item)
▼VIP_99a05c35-1194-4272-802c-56da372632e1	Dictionary	(3 items)
▼value	Dictionary	(3 items)
n	String	Rob Lee
v	Number	1
▼a	Array	(1 item)
Item 0	String	rlee@sans.org
remotevalue	Data	<018ab70b 7cab0c18 00000000 62706c69 73743030 d3010203 0405
timestamp	Number	403,483,516



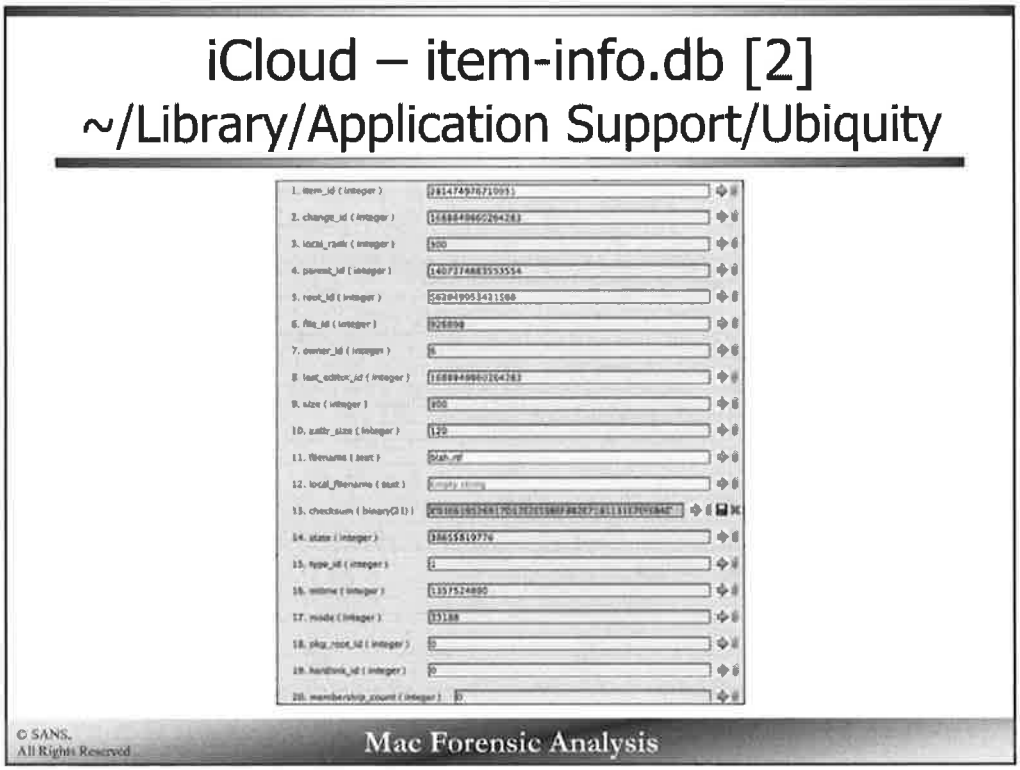
The top screenshot shows a tree command output of the `~/Library/Application Support/Ubiquity/` directory. Each peer directory contains a chunk store database (more on that later when we talk about Versions), configuration data files and a SQLite database, `item-info.db`. This database contains details about the synced documents and the device with which they are synced. The `config` file contains the iCloud person ID.

The bottom screenshot shows an example of the `peer_names` table. This table contains all the iCloud peers or devices that are configured to sync with a particular iCloud account. Of particular interest is that this database also contains peers that are no longer in use – it appears that it keeps a history of synced device names!

```
nibble:Ubiquity sledwards$ tree -a .
.
└── peer-195BBE38-4914-EEE3-2435-97F111CCB3F3-v23
    ├── .cs
    │   ├── ChunkStoreDatabase
    │   └── ChunkStoreDatabase-wal
    ├── config
    ├── control-even
    ├── control-odd
    ├── item-info.db
    └── path_rules_v1
```

TABLE peer_names Search Show All

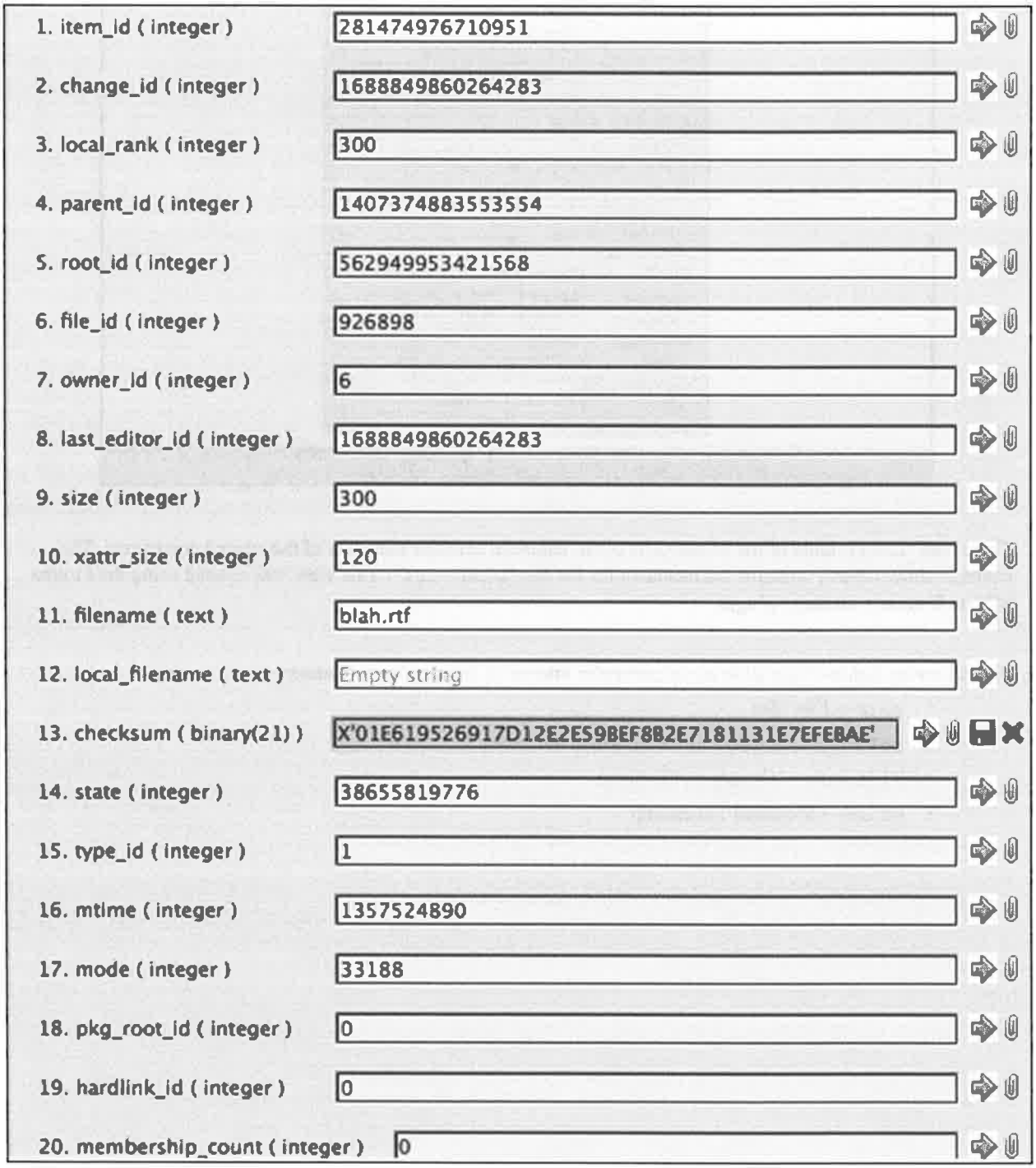
rowid	uuid	name	change_id
2	6386E206-ABC5-41DB-82C4-13765DD7C681	iCloud	0
3	FE881D1C-A380-1210-D99A-9BC641281647	bit	0
4	BE08C420-45A5-F5F9-BDAB-B46B002C738D	byte	0
8	F743848C-D702-915F-1525-D04F447AD4BD	miPhone4S	0
9	384B6D45-AEAF-182A-EC83-368B6FFB08DA	nibble.blah	1407374883553624
10	F2265FCF-A0A5-F495-372A-6E0E898C5698	bit	1688849860264194
12	566EFA6E-7A5B-4D67-9838-31AAEA4D568B	miPhone4S	2533274790396162
13	AED73627-CC8E-452A-A6D0-0A7A54EBA3F9	miPhone5s	2814749767106818
18	195BBE38-4914-EEE3-2435-97F111CCB3F3	nibble	281474976711277
19	DC711849-0C49-C819-8E70-75E417856547	ellingson	1970324836974950

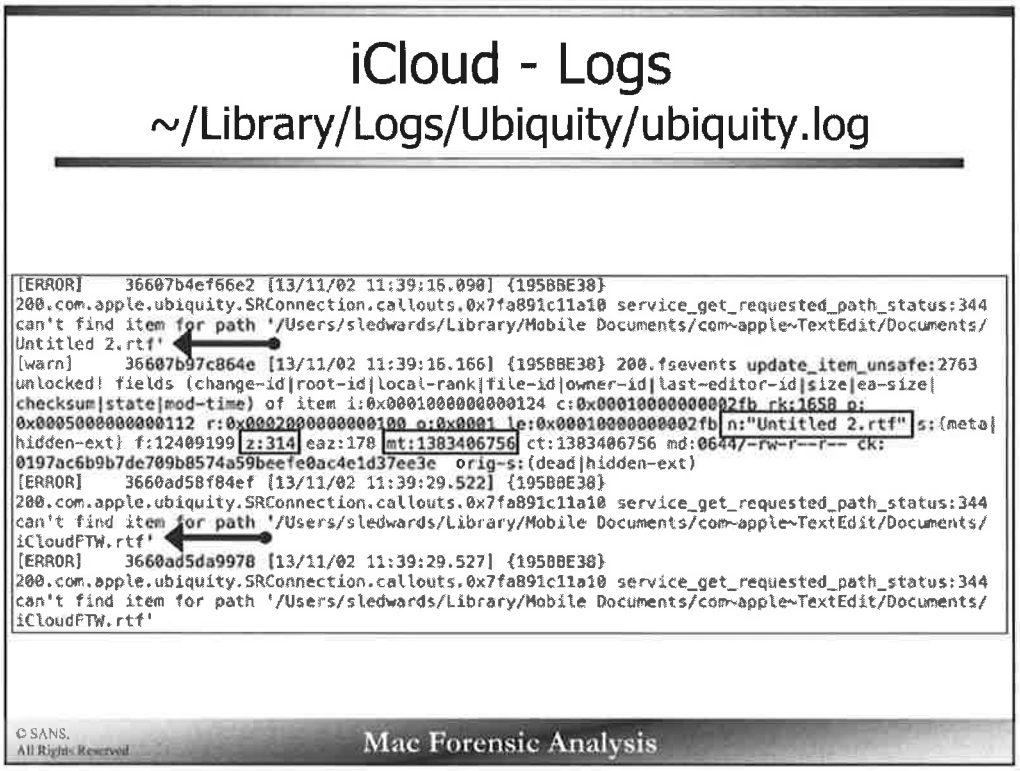


The `item_table` table of the `item-info.db` database contains metadata of the synced documents. The example shown above contains the metadata for the file "`blah.rtf`". This view was created using the Firefox SQLite Database manager plugin.

The following fields of this table are of particular interest to forensic investigators:

- `size` – File Size
- `xattr_size` – Size of the extended attributes
- `filename` – Document Filename
- `mtime` – Modified Timestamp


1. item_id (integer)
281474976710951
2. change_id (integer)
1688849860264283
3. local_rank (integer)
300
4. parent_id (integer)
1407374883553554
5. root_id (integer)
562949953421568
6. file_id (integer)
926898
7. owner_id (integer)
6
8. last_editor_id (integer)
1688849860264283
9. size (integer)
300
10. xattr_size (integer)
120
11. filename (text)
blah.rtf
12. local_filename (text)
Empty string
13. checksum (binary(21))
X'01E619526917D12E2E59BEF8B2E7181131E7EFEBAE'
14. state (integer)
38655819776
15. type_id (integer)
1
16. mtime (integer)
1357524890
17. mode (integer)
33188
18. pkg_root_id (integer)
0
19. hardlink_id (integer)
0
20. membership_count (integer)
0



iCloud also keeps log files in the `~/Library/Logs/Ubiquity/` directory. The `ubiquity.log` file contains on-disk file paths to synced documents with associated metadata.

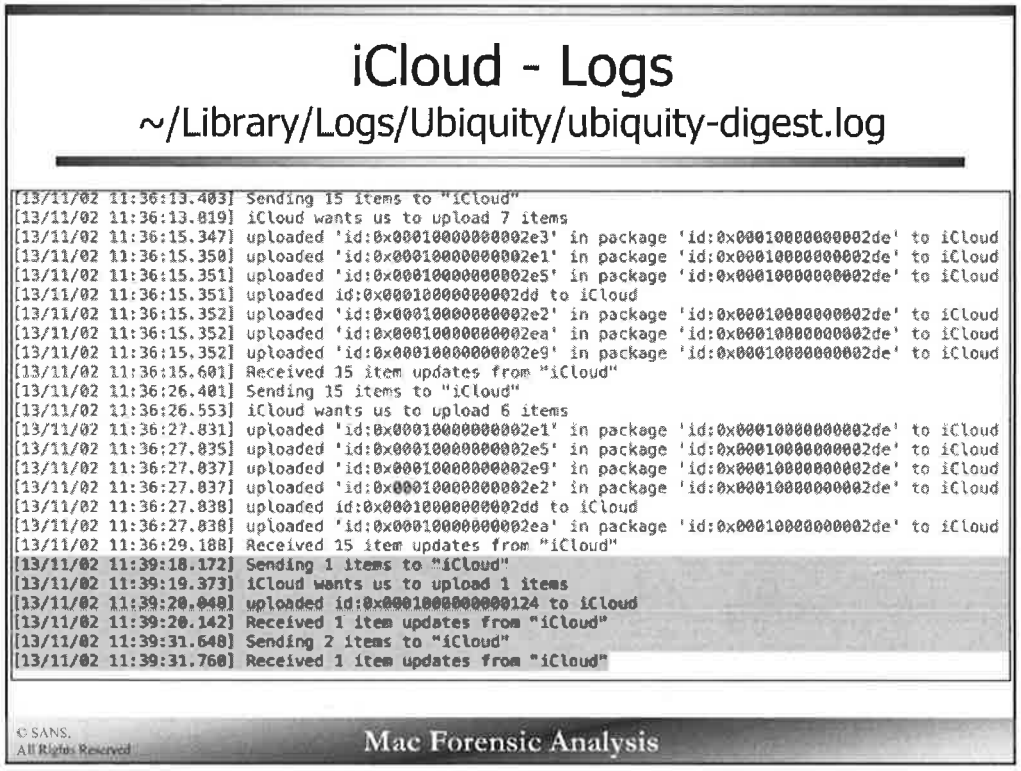
The example in the screenshot provided shows one RTF file that started with the name "`Untitled 2.rtf`", and was saved with another filename "`iCloudFTW.rtf`". The metadata attributes are named with a shortened pattern, of note are the following file attributes:

- `n` – Filename
- `z` – File Size
- `mt` – Modified Time
- `ct` – Create Time
- `md` - Metadata

These logs can allow an investigator to discover other documents that may be of importance to them and when they were created and synced.

```
[ERROR]    36607b4ef66e2 [13/11/02 11:39:16.090] {195BBE38}
200.com.apple.ubiquity.SRConnection.callouts.0x7fa891c11a10 service_get_requested_path_status:344
can't find item for path '/Users/sledwards/Library/Mobile Documents/com~apple~TextEdit/Documents/
Untitled 2.rtf'
[warn]     36607b97c864e [13/11/02 11:39:16.166] {195BBE38} 200.fsevents update_item_unsafe:2763
unlocked! fields (change-id|root-id|local-rank|file-id|owner-id|last-editor-id|size|ea-size|
checksum|state|mod-time) of item i:0x0001000000000124 c:0x00010000000002fb rk:1658 p:
0x0005000000000112 r:0x0002000000000100 o:0x0001 le:0x00010000000002fb n:"Untitled 2.rtf" s:(meta|
hidden-ext) f:12409199 z:314 eaz:178 mt:1383406756 ct:1383406756 md:0644/-rw-r--r-- ck:
0197ac6b9b7de709b8574a59beefe0ac4e1d37ee3e  orig-s:(dead|hidden-ext)
[ERROR]    3660ad58f84ef [13/11/02 11:39:29.522] {195BBE38}
200.com.apple.ubiquity.SRConnection.callouts.0x7fa891c11a10 service_get_requested_path_status:344
can't find item for path '/Users/sledwards/Library/Mobile Documents/com~apple~TextEdit/Documents/
iCloudFTW.rtf'
[ERROR]    3660ad5da9978 [13/11/02 11:39:29.527] {195BBE38}
200.com.apple.ubiquity.SRConnection.callouts.0x7fa891c11a10 service_get_requested_path_status:344
can't find item for path '/Users/sledwards/Library/Mobile Documents/com~apple~TextEdit/Documents/
iCloudFTW.rtf'
```

160

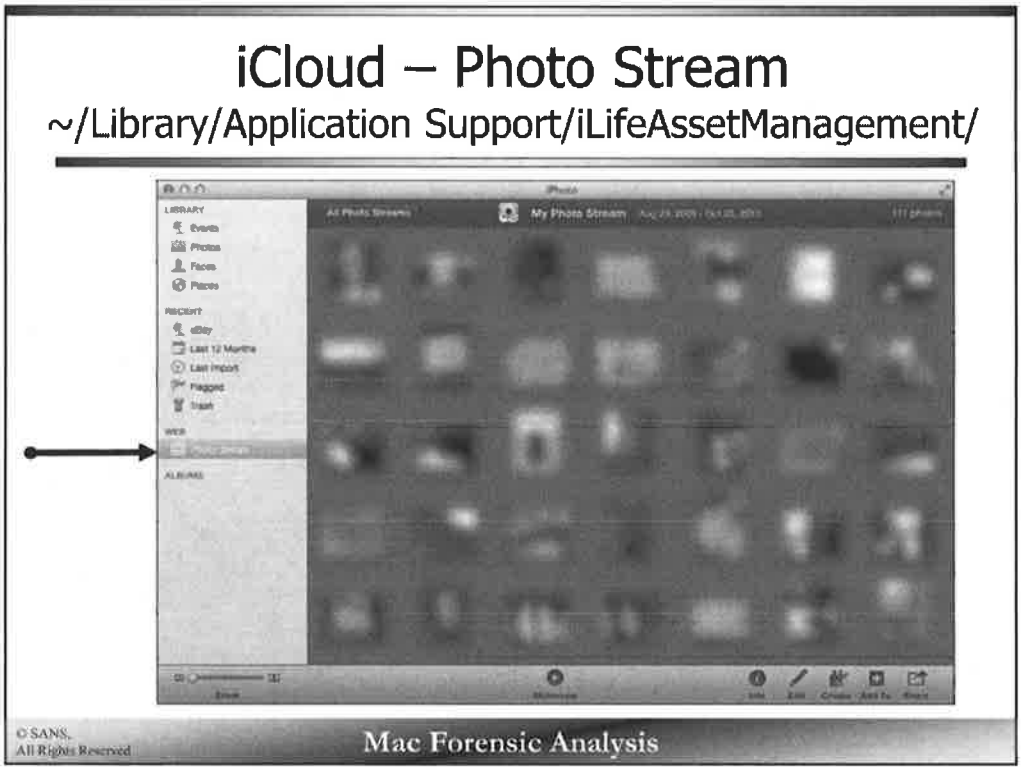


Another log located in the `~/Library/Logs/Ubiquity/` directory, is the `ubiquity-digest.log`. This log file shows when files are synced to and from the iCloud servers.

The highlighted section (in blue) is congruent syncing of the iCloudFTW.rtf file used in the previous slide example. We'll have to do some timeline analysis to determine what file was uploaded (or which files were downloaded) from the iCloud servers.

The non-highlighted log entries show a Pages document being uploaded to the servers. The amount of items uploaded is far more than the single RTF file – this is because a Pages document is a package file. It contains multiple files in one Pages document, therefore it must upload multiple files. This distinction can give an examiner an idea of what might have been uploaded at a certain time.

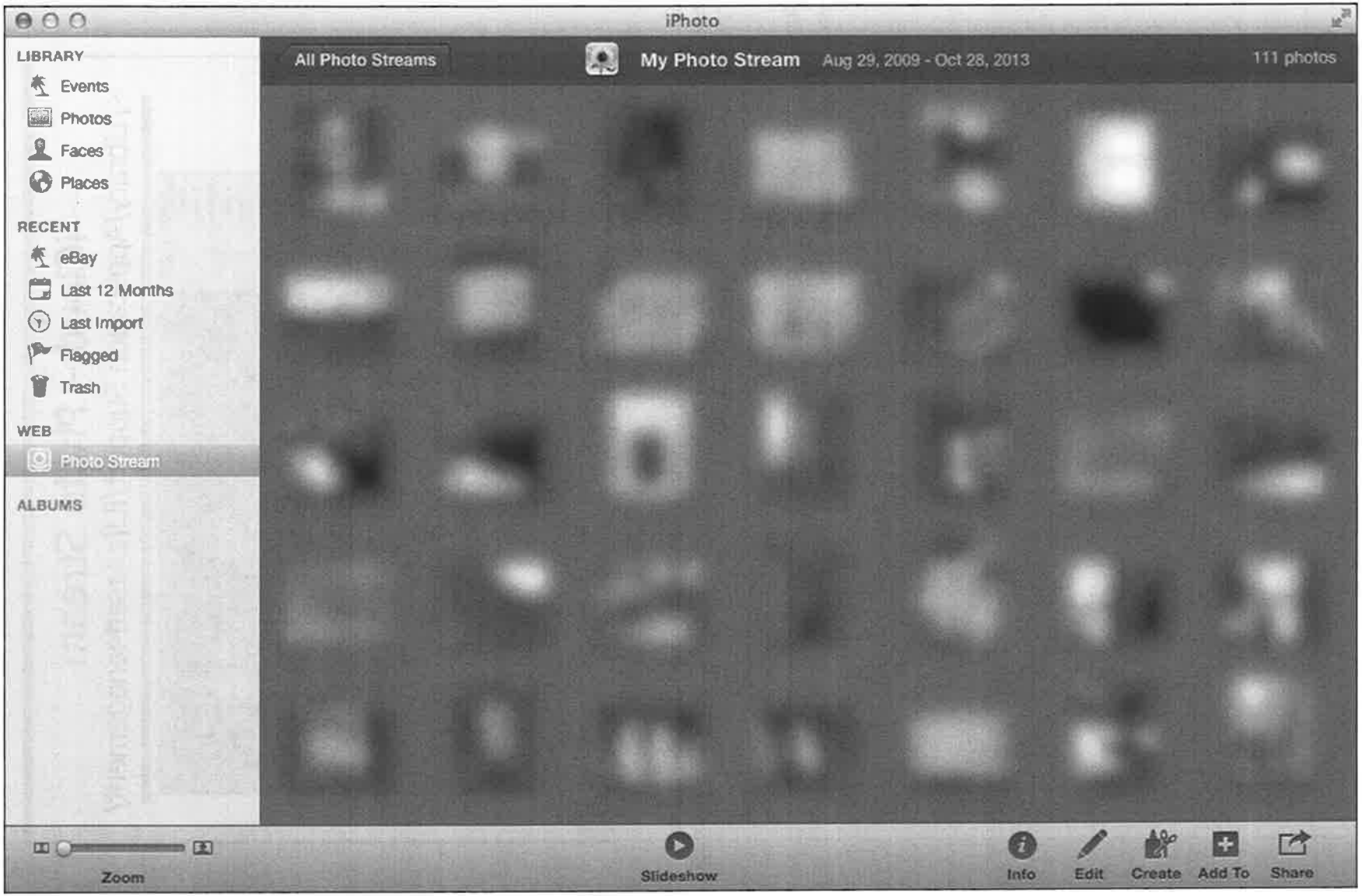
```
[13/11/02 11:36:13.403] Sending 15 items to "iCloud"
[13/11/02 11:36:13.819] iCloud wants us to upload 7 items
[13/11/02 11:36:15.347] uploaded 'id:0x00010000000002e3' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:15.350] uploaded 'id:0x00010000000002e1' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:15.351] uploaded 'id:0x00010000000002e5' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:15.351] uploaded id:0x00010000000002dd to iCloud
[13/11/02 11:36:15.352] uploaded 'id:0x00010000000002e2' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:15.352] uploaded 'id:0x00010000000002ea' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:15.352] uploaded 'id:0x00010000000002e9' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:15.601] Received 15 item updates from "iCloud"
[13/11/02 11:36:26.401] Sending 15 items to "iCloud"
[13/11/02 11:36:26.553] iCloud wants us to upload 6 items
[13/11/02 11:36:27.831] uploaded 'id:0x00010000000002e1' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:27.835] uploaded 'id:0x00010000000002e5' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:27.837] uploaded 'id:0x00010000000002e9' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:27.837] uploaded 'id:0x00010000000002e2' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:27.838] uploaded id:0x00010000000002dd to iCloud
[13/11/02 11:36:27.838] uploaded 'id:0x00010000000002ea' in package 'id:0x00010000000002de' to iCloud
[13/11/02 11:36:29.188] Received 15 item updates from "iCloud"
[13/11/02 11:39:18.172] Sending 1 items to "iCloud"
[13/11/02 11:39:19.373] iCloud wants us to upload 1 items
[13/11/02 11:39:20.048] uploaded id:0x0001000000000124 to iCloud
[13/11/02 11:39:20.142] Received 1 item updates from "iCloud"
[13/11/02 11:39:31.648] Sending 2 items to "iCloud"
[13/11/02 11:39:31.760] Received 1 item updates from "iCloud"
```

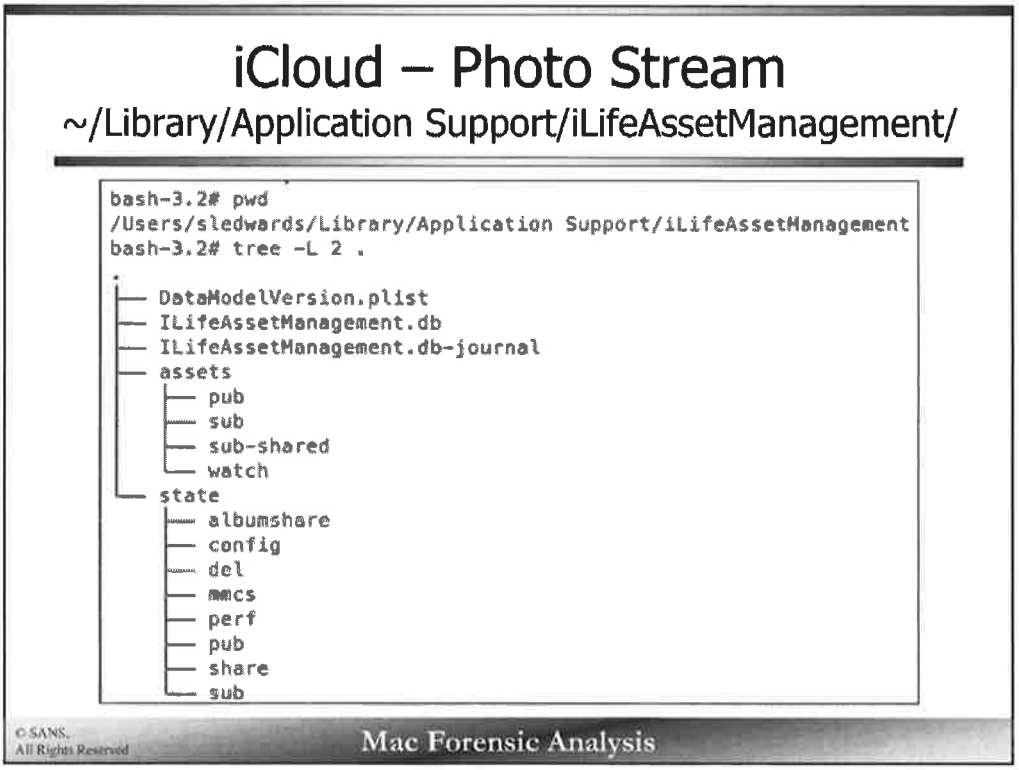


Photos taken with iDevices and synced with iCloud are called a users Photo Stream. The Photo Stream is updated in near real-time and can be shared with other users. A user can access their Photo Stream by clicking on "Photo Stream" in the iPhoto sidebar (as shown above).

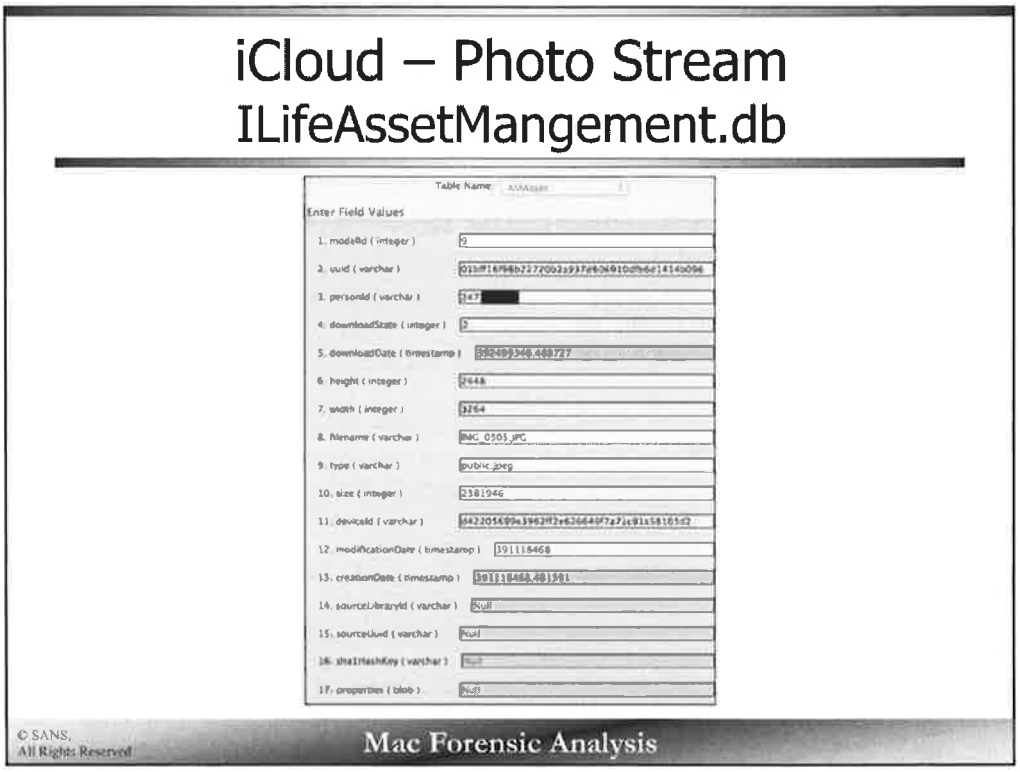
References:

http://www.apple.com/icloud/icloud-photo-sharing.html


iPhoto
LIBRARY
Events
Photos
Faces
Places
RECENT
eBay
Last 12 Months
Last Import
Flagged
Trash
WEB
Photo Stream
ALBUMS
All Photo Streams
My Photo Stream
Aug 29, 2009 - Oct 28, 2013
111 photos
Zoom
Slideshow
Info
Edit
Create
Add To
Share



The Photo Stream photos are saved in the `~/Library/Application Support/iLifeAssetManagement/` directory. In the `tree` command output above, we can see this directory contains a database and an `assets` directory.



The `ILifeAssetManagement.db` SQLite database file contains a table, `AMAsset`, that stores the metadata for each photo in the Photo Stream.

The screenshot above of an entry in this table (using the Firefox SQLite manager plugin) details various metadata attributes for one photo, "`IMG_0505.JPG`".

- `personid` – iCloud Person ID
- `downloadDate` – When the file was downloaded from the iCloud servers
- `height & width` – pixel height/width of the photo
- `filename` – Photo Filename
- `size` – File Size
- `deviceid` – Device UUID of the device that took the picture
- `modificationDate & creationDate` – When the photo was last modified (i.e., taken with the iDevice)

Analyzing the `height` and `width` columns of this table can show some interesting pattern. You can quickly tell what files are photos (taken with the camera) and "screenshots" saved by the iDevice.

Reviewing the `deviceid` column can show you when a new phone or iDevice was purchased and when it was being used. It may also tell you if the target was using multiple devices at once. Remember the `deviceID` is the UUID of the iDevice, this UUID is unique across all iDevices ever made. You can easily correlate when a photo of interest was taken with a particular iPhone, iPod, or iPad (assuming you know this device's UUID).

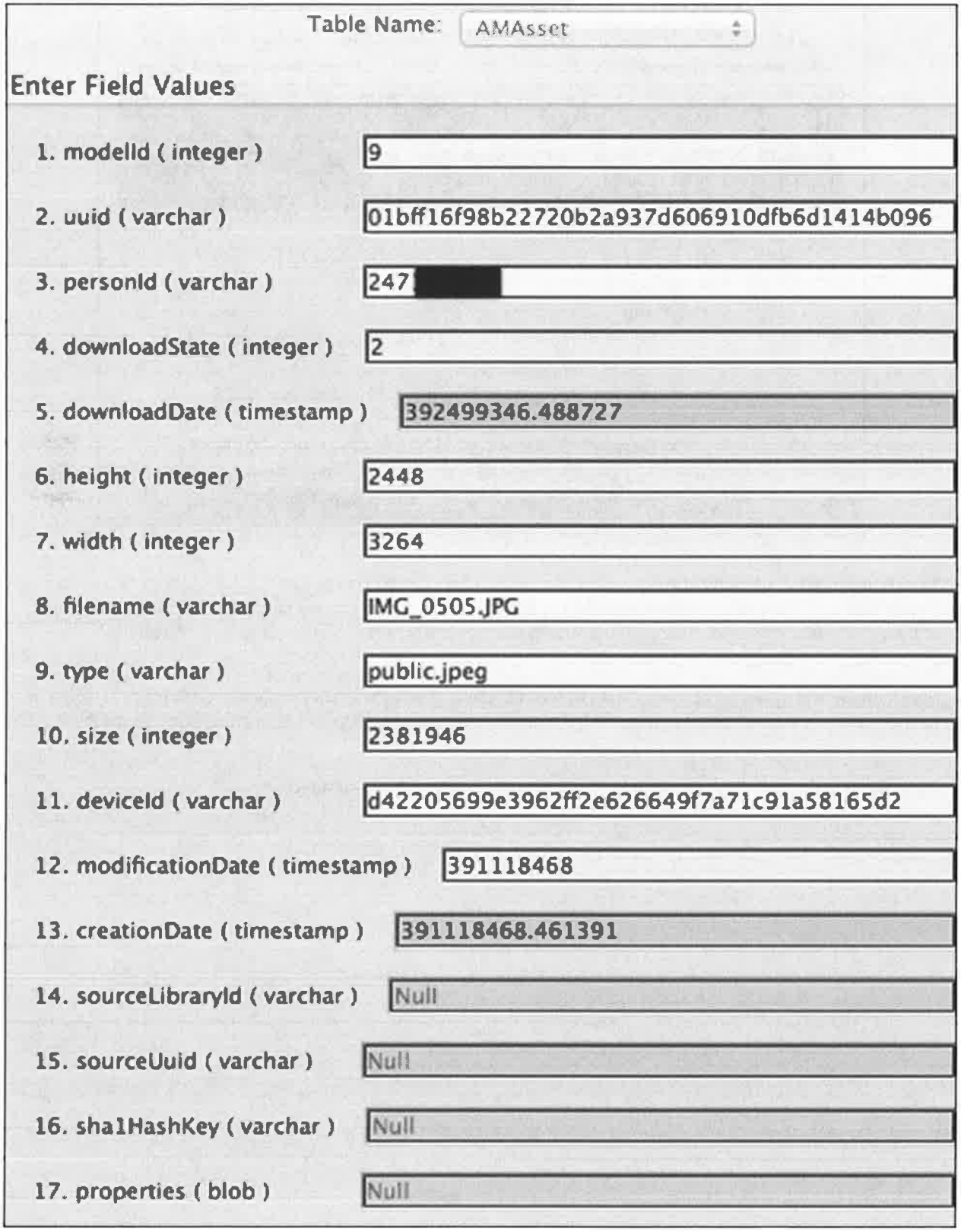
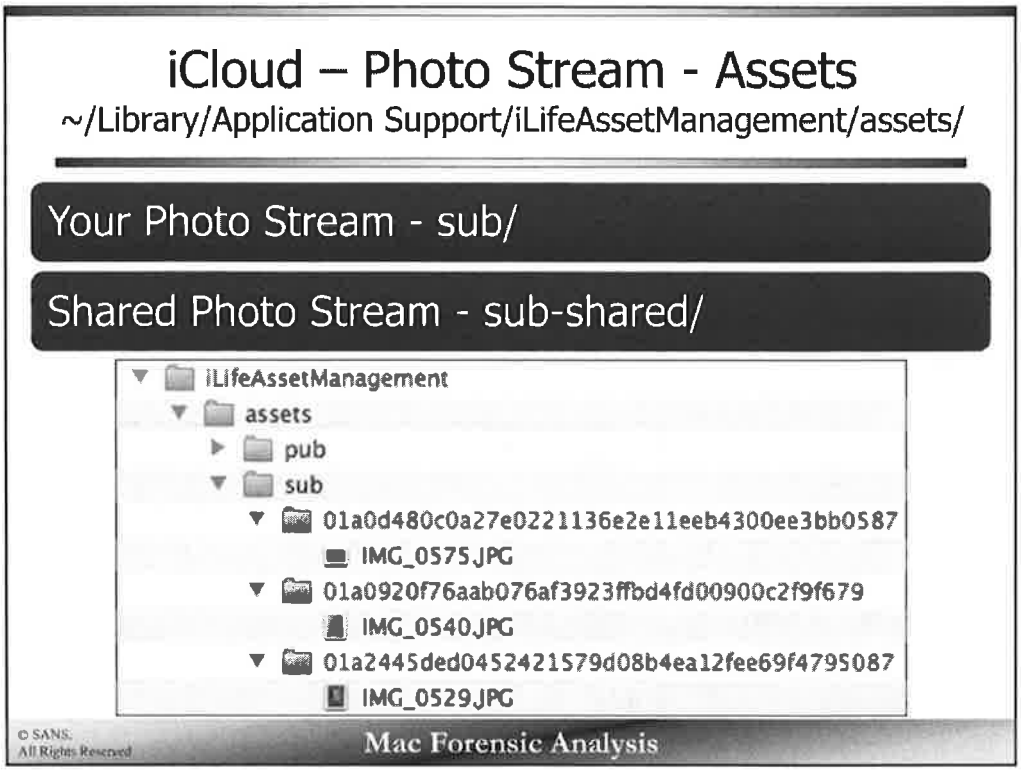


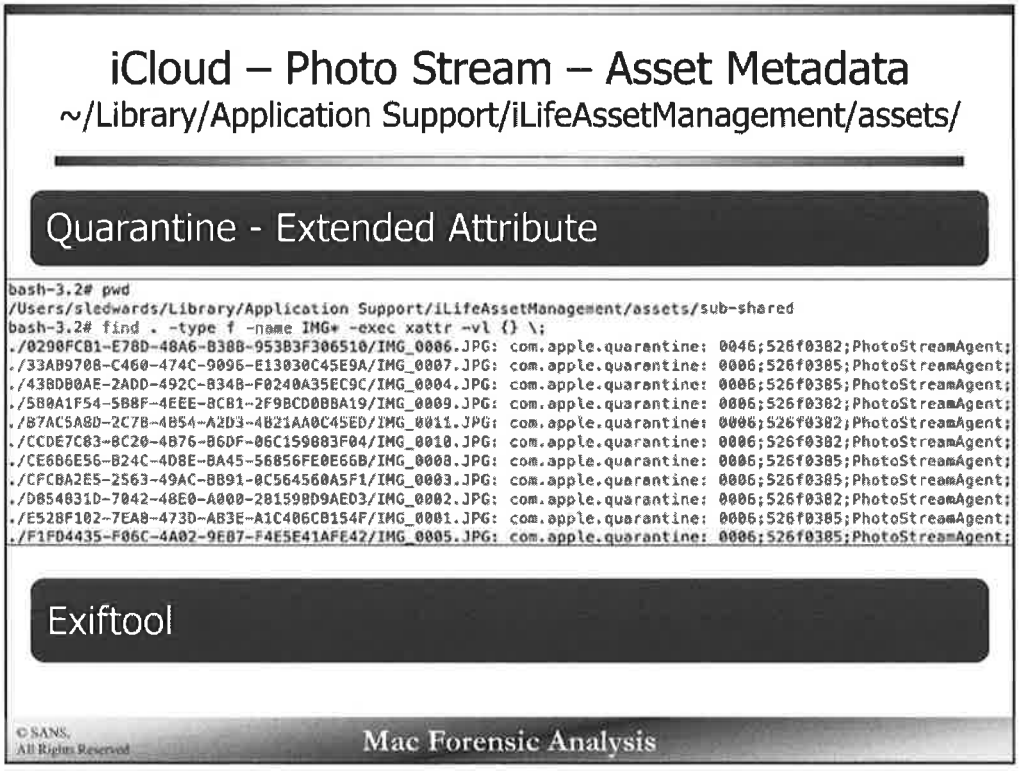
Table Name: AMAsset
Enter Field Values
1. modelId (integer) 9
2. uuid (varchar) 01bff16f98b22720b2a937d606910dfb6d1414b096
3. personId (varchar) 247
4. downloadState (integer) 2
5. downloadDate (timestamp) 392499346.488727
6. height (integer) 2448
7. width (integer) 3264
8. filename (varchar) IMG_0505.JPG
9. type (varchar) public.jpeg
10. size (integer) 2381946
11. deviceId (varchar) d42205699e3962ff2e626649f7a71c91a58165d2
12. modificationDate (timestamp) 391118468
13. creationDate (timestamp) 391118468.461391
14. sourceLibraryId (varchar) Null
15. sourceUuid (varchar) Null
16. sha1HashKey (varchar) Null
17. properties (blob) Null



The photos in the Photo Stream are stored in the `~/Library/Application Support/iLifeAssetManagement/assets/` directory. A user's own photo stream photos are located in the `/sub` directory, while shared photo stream photos are stored in the `sub-shared/` directory.

Shown in the screenshot is the `/sub` directory containing a user's photo stream photos. Each photo is stored in its own directory named after the photo's UUID, as found in the `iLifeAssetManagement.db` database.

Each photo in the `sub-shared` directory is contained in a GUID named directory.



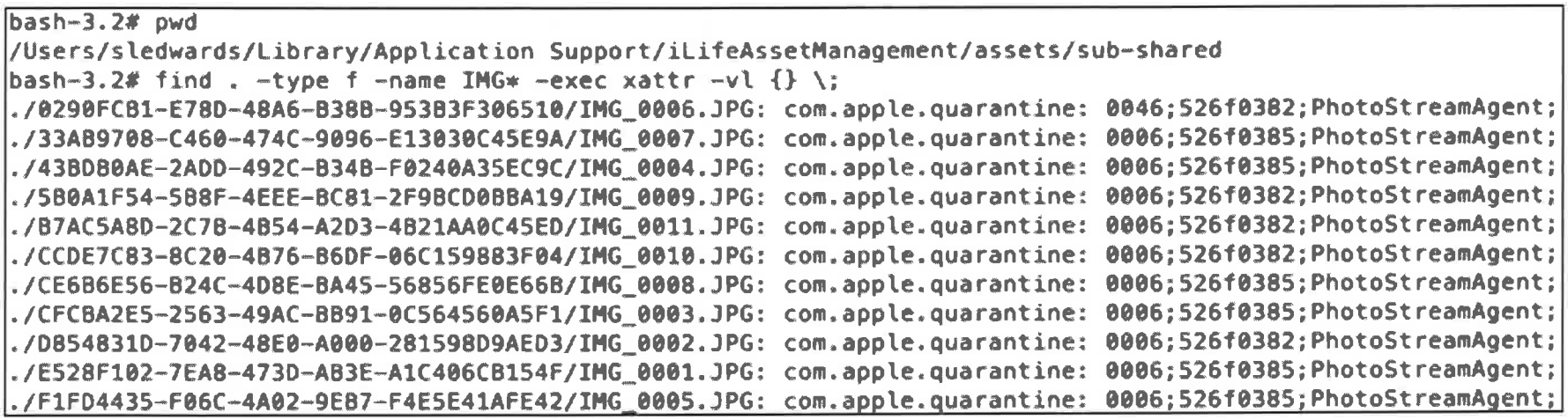
Each photo in `iLifeAssetManagemet` sub-directories has a `com.apple.quarantine` extended attribute.

The output from the `xattr` command shown above demonstrates that the quarantine extended attribute came from the `PhotoStreamAgent` program.

Extended attributes are not the only metadata associated with these photos, there is always the photo EXIF data in the photo file itself. We can use the tool `exiftool` to print out a listing of additional metadata, available here: http://www.sno.phy.queensu.ca/~phil/exiftool/

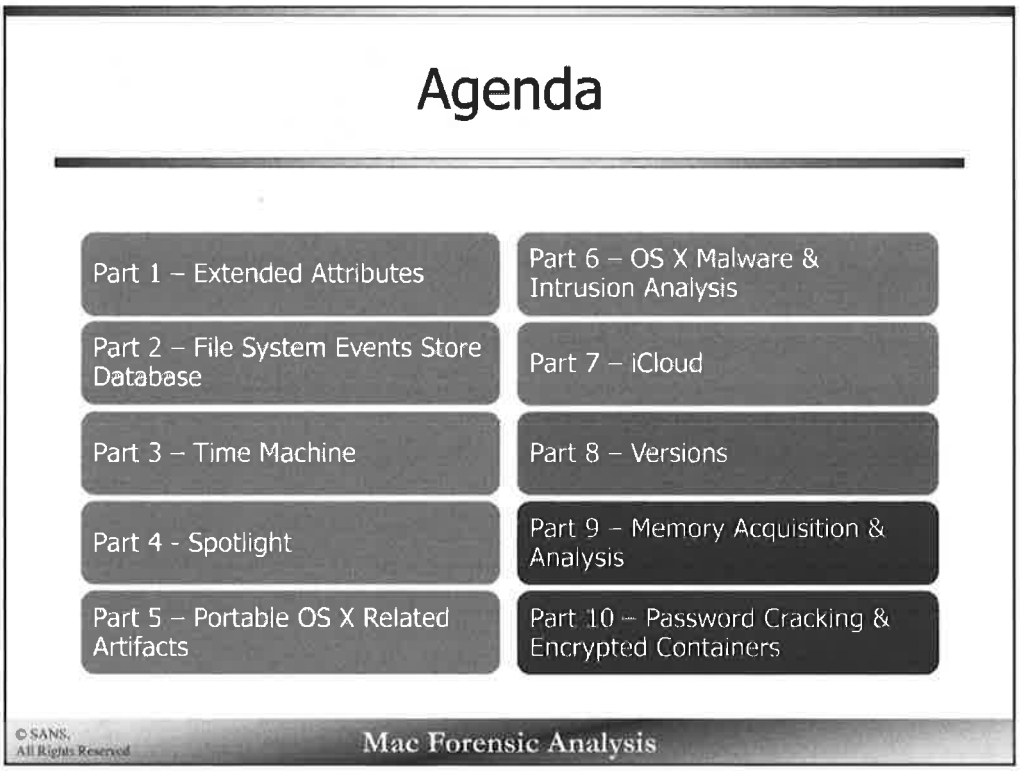
The next few pages contain example `exiftool` output. Take note of the following fields:

- `Make` – iDevice Make (Apple)
- `Model Camera Name` – iDevice Model
- `Software` – iOS Software Version
- `Lens Model` – Back/Front Camera
- Timestamps –
    - `File Modification Date/Time`
    - `File Access Date/Time`
    - `File Inode Change Date/Time`
    - `Date/Time Original`
    - `Create Date`

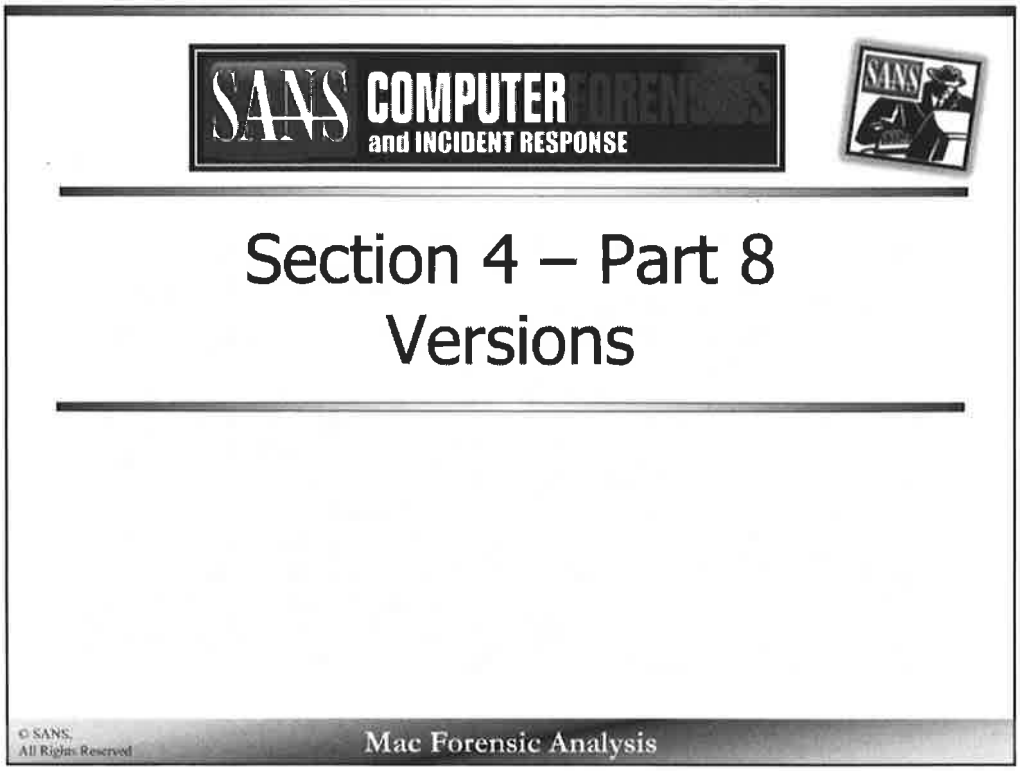

bash-3.2# pwd
/Users/sledwards/Library/Application Support/iLifeAssetManagement/assets/sub-shared
bash-3.2# find . -type f -name IMG* -exec xattr -vl {} \;
./0290FCB1-E78D-48A6-B38B-953B3F306510/IMG_0006.JPG: com.apple.quarantine: 0046;526f0382;PhotoStreamAgent;
./33AB9708-C460-474C-9096-E13030C45E9A/IMG_0007.JPG: com.apple.quarantine: 0006;526f0385;PhotoStreamAgent;
./43BD80AE-2ADD-492C-B34B-F0240A35EC9C/IMG_0004.JPG: com.apple.quarantine: 0006;526f0385;PhotoStreamAgent;
./5B0A1F54-588F-4EEE-BC81-2F9BCD0BBA19/IMG_0009.JPG: com.apple.quarantine: 0006;526f0382;PhotoStreamAgent;
./B7AC5A8D-2C7B-4B54-A2D3-4B21AA0C45ED/IMG_0011.JPG: com.apple.quarantine: 0006;526f0382;PhotoStreamAgent;
./CCDE7C83-8C20-4B76-B6DF-06C1598B3F04/IMG_0010.JPG: com.apple.quarantine: 0006;526f0382;PhotoStreamAgent;
./CE6B6E56-B24C-4D8E-BA45-56856FE0E66B/IMG_0008.JPG: com.apple.quarantine: 0006;526f0385;PhotoStreamAgent;
./CFCBA2E5-2563-49AC-BB91-0C564560A5F1/IMG_0003.JPG: com.apple.quarantine: 0006;526f0385;PhotoStreamAgent;
./D854831D-7042-48E0-A000-281598D9AED3/IMG_0002.JPG: com.apple.quarantine: 0006;526f0382;PhotoStreamAgent;
./E528F102-7EA8-473D-AB3E-A1C406CB154F/IMG_0001.JPG: com.apple.quarantine: 0006;526f0385;PhotoStreamAgent;
./F1FD4435-F06C-4A02-9EB7-F4E5E41AFE42/IMG_0005.JPG: com.apple.quarantine: 0006;526f0385;PhotoStreamAgent;

```
bash-3.2# exiftool IMG_0006.JPG
ExifTool Version Number         : 9.39
File Name                       : IMG_0006.JPG
Directory                       : .
File Size                       : 718 kB
File Modification Date/Time     : 2013:10:28 20:38:28-04:00
File Access Date/Time           : 2013:11:03 05:55:17-05:00
File Inode Change Date/Time     : 2013:11:03 05:55:16-05:00
File Permissions                : rw-------
File Type                       : JPEG
MIME Type                       : image/jpeg
Exif Byte Order                 : Big-endian (Motorola, MM)
Make                            : Apple
Camera Model Name               : iPhone 5s
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Software                        : 7.0.2
Modify Date                     : 2013:10:28 20:15:51
Y Cb Cr Positioning             : Centered
Exposure Time                   : 1/30
F Number                        : 2.2
Exposure Program                : Program AE
ISO                             : 250
Exif Version                    : 0221
Date/Time Original              : 2013:10:28 20:15:51
Create Date                     : 2013:10:28 20:15:51
Components Configuration        : Y, Cb, Cr, -
Shutter Speed Value             : 1/30
Aperture Value                  : 2.2
Brightness Value                : 1.757035003
Metering Mode                   : Multi-segment
Flash                           : No Flash
Focal Length                    : 4.1 mm
Subject Area                    : 1631 1223 1795 1077
Run Time Flags                  : Valid
Run Time Value                  : 10637257523333
Run Time Scale                  : 1000000000
Run Time Epoch                  : 0
Sub Sec Time Original           : 446
Sub Sec Time Digitized          : 446
```

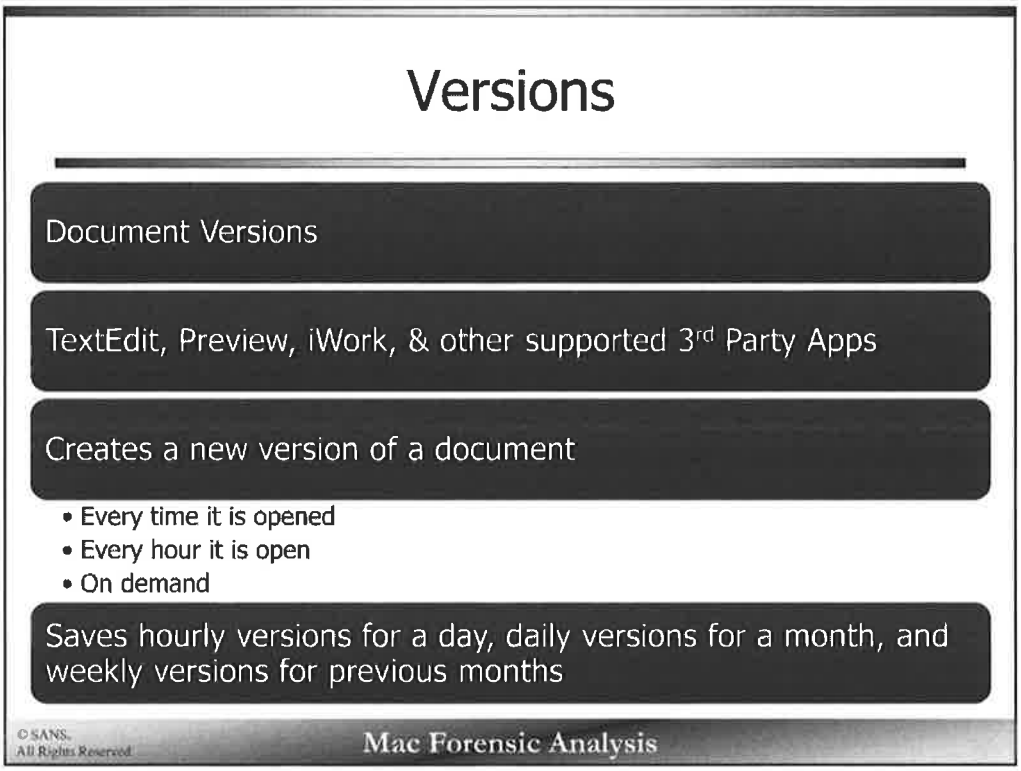
Flashpix Version	: 0100
Color Space	: sRGB
Exif Image Width	: 3264
Exif Image Height	: 2448
Sensing Method	: One-chip color area
Scene Type	: Directly photographed
Exposure Mode	: Auto
White Balance	: Auto
Focal Length In 35mm Format	: 30 mm
Scene Capture Type	: Standard
Lens Info	: 4.12mm f/2.2
Lens Make	: Apple
Lens Model	: iPhone 5s back camera 4.12mm f/2.2
Image Width	: 2048
Image Height	: 1536
Encoding Process	: Baseline DCT, Huffman coding
Bits Per Sample	: 8
Color Components	: 3
Y Cb Cr Sub Sampling	: YCbCr4:2:0 (2 2)
Aperture	: 2.2
Image Size	: 2048x1536
Run Time Since Power Up	: 2:57:17
Scale Factor To 35 mm Equivalent	: 7.3
Shutter Speed	: 1/30
Create Date	: 2013:10:28 20:15:51.446
Date/Time Original	: 2013:10:28 20:15:51.446
Circle Of Confusion	: 0.004 mm
Field Of View	: 61.9 deg
Focal Length	: 4.1 mm (35 mm equivalent: 30.0 mm)
Hyperfocal Distance	: 1.87 m
Light Value	: 5.9



This page intentionally left blank.



This page intentionally left blank.



Versions was introduced in 10.7 as a way for the OS X system to automatically create document versions and backup copies of certain types of documents for the users. Ideally, this is meant as an easy way for a user to revert back to a previous version of the document, or to restore a document after a system crash.

It will periodically create a new version of the document. For example, it will create one every time the document is opened, every hour after it is open, and when the user decides. The hourly document versions will be saved for a day, daily versions will be saved for a month, and weekly versions for the previous months.

Only certain programs on the Mac support this feature including:

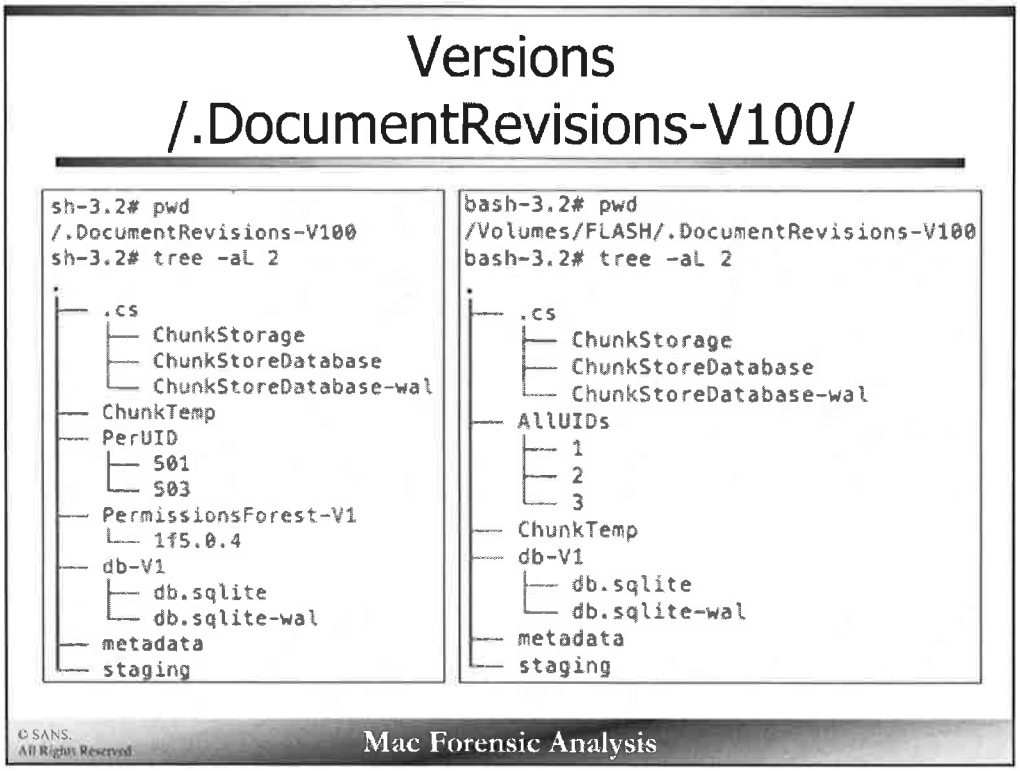
- TextEdit
- iWork Applications (Numbers, Keynote, Pages)
- Preview

Other third-party apps may also support this; however, it is not mandatory. Microsoft Office does not implement it as it uses its own version of auto save.

References:

http://support.apple.com/kb/HT4753

http://support.apple.com/kb/PH11444



Versions allows documents and files to have multiple "generations". Only one file for each document actually exists on the system, while changes are found in Chunk Storage (discussed later).

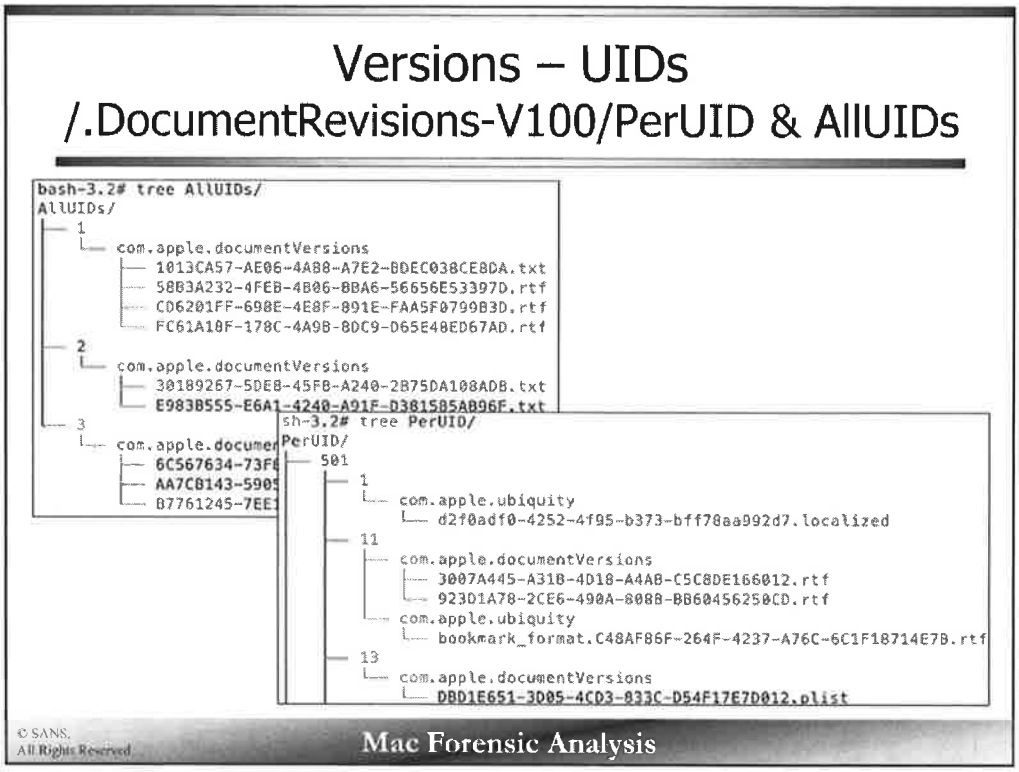
Each volume will contain a hidden directory called `.DocumentRevisions-V100`. Depending on the type of volume the contents of this directory may be different.

The example on the left is from a system volume with a bootable version of OS X, while the example on the right is one from a USB drive. The major difference between the two examples is the `PerUID/AllUIDs` directories (and the existence of the `PermissionsForest-V1` directory - which may or may not be present).

The system volume allows document revisions per each user under their specific user ID (501, 503, etc.), while the USB drive lumps them all under AllUIDs.

```
sh-3.2# pwd
/.DocumentRevisions-V100
sh-3.2# tree -aL 2
.
├── .cs
│   ├── ChunkStorage
│   ├── ChunkStoreDatabase
│   └── ChunkStoreDatabase-wal
├── ChunkTemp
├── PerUID
│   ├── 501
│   └── 503
├── PermissionsForest-V1
│   └── 1f5.0.4
├── db-V1
│   ├── db.sqlite
│   └── db.sqlite-wal
├── metadata
└── staging
```

```
bash-3.2# pwd
/Volumes/FLASH/.DocumentRevisions-V100
bash-3.2# tree -aL 2
.
├── .cs
│   ├── ChunkStorage
│   ├── ChunkStoreDatabase
│   └── ChunkStoreDatabase-wal
├── AllUIDs
│   ├── 1
│   ├── 2
│   └── 3
├── ChunkTemp
├── db-V1
│   ├── db.sqlite
│   └── db.sqlite-wal
├── metadata
└── staging
```



The top screenshot shows an example of a USB `AllUIDs` directory, while the bottom screenshot shows a system volume's `PerUID` directory (partial contents for brevity).

Under the `AllUIDs` or `PerUID/<UID>` directories, there may be a number of directories with hex filenames. This numbered directory starts at 1 and counts sequentially in hex numbers (i.e., 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, 10, 11, 12, 13, 14…). These directory names are unique across all UIDs on system volumes.

Under the hex numbered directories are other directories named in reverse DNS format:

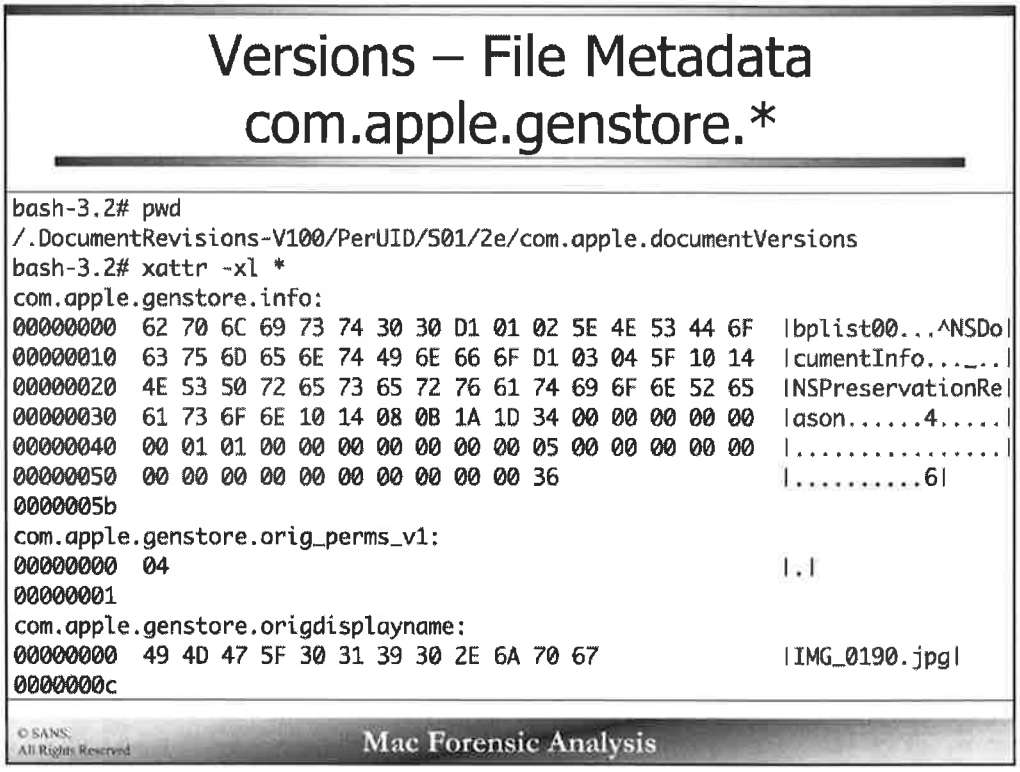
- `com.apple.documentVersions` – Documents that are saved on the local volume
- `com.apple.ubiquity` – Documents that are saved on the local volume & iCloud

Each document "generation" is saved with a different GUID filename with the original file extension in a number directory.

Filenames under the `com.apple.ubiquity` directory will have the base filename before the GUID.

```
bash-3.2# tree AllUIDs/
AllUIDs/
├── 1
│   └── com.apple.documentVersions
│       ├── 1013CA57-AE06-4AB8-A7E2-BDEC038CE8DA.txt
│       ├── 58B3A232-4FEB-4B06-BBA6-56656E53397D.rtf
│       ├── CD6201FF-698E-4E8F-891E-FAA5F0799B3D.rtf
│       └── FC61A18F-178C-4A9B-8DC9-D65E48ED67AD.rtf
├── 2
│   └── com.apple.documentVersions
│       ├── 30189267-5DE8-45F8-A240-2B75DA108ADB.txt
│       └── E983B555-E6A1-4240-A91F-D381585AB96F.txt
└── 3
    └── com.apple.documentVersions
        ├── 6C567634-73F8-4803-AD96-30876E416FEE.rtf
        ├── AA7CB143-5905-48AF-8549-D0C6BBA12542.rtf
        └── B7761245-7EE1-43E5-AC84-54E67C31DDB6.rtf
```

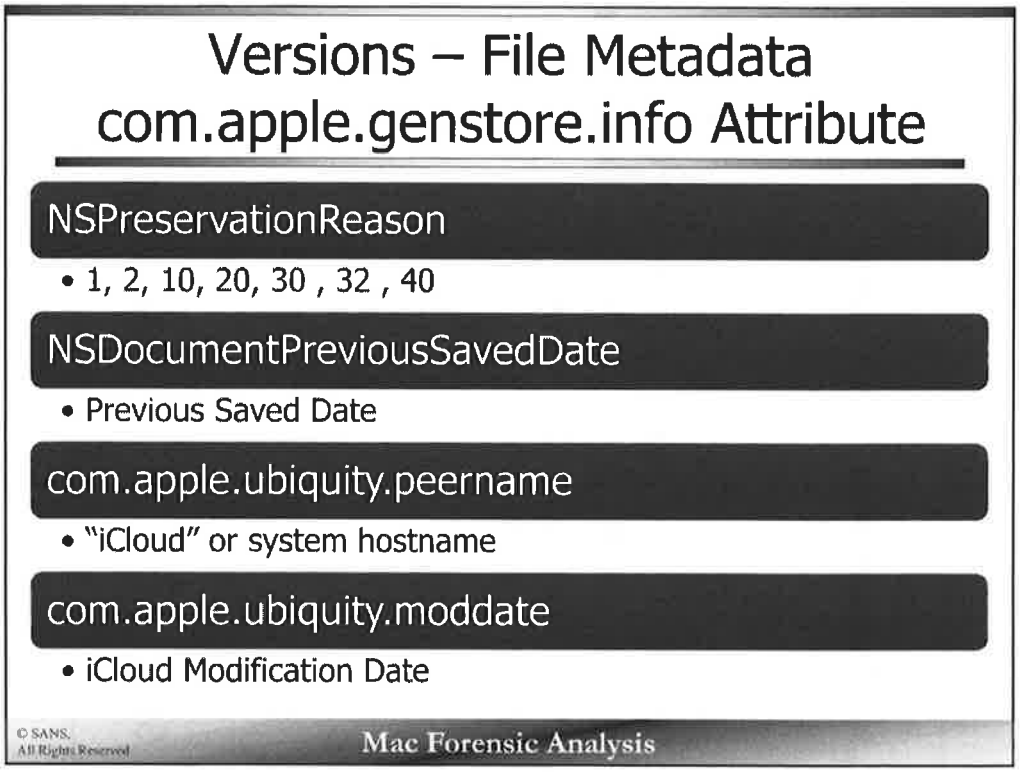
```
sh-3.2# tree PerUID/
PerUID/
└── 501
    ├── 1
    │   └── com.apple.ubiquity
    │       └── d2f0adf0-4252-4f95-b373-bff78aa992d7.localized
    ├── 11
    │   ├── com.apple.documentVersions
    │   │   ├── 3007A445-A31B-4D18-A4AB-C5C8DE166012.rtf
    │   │   └── 923D1A78-2CE6-490A-8088-BB60456250CD.rtf
    │   └── com.apple.ubiquity
    │       └── bookmark_format.C48AF86F-264F-4237-A76C-6C1F18714E7B.rtf
    └── 13
        └── com.apple.documentVersions
            └── DBD1E651-3D05-4CD3-833C-D54F17E7D012.plist
```



Each generation file has extended attributes associated with "genstore" or generational storage.

- `com.apple.genstore.origdisplayname` – Filename associated with this generation of the file.
- `com.apple.genstore.orig_perms_v1` – Examples that I've seen only have 0x04 or 0x1C as its contents.
- `com.apple.genstore.info` – Embedded binary property list file.

```
bash-3.2# pwd
/.DocumentRevisions-V100/PerUID/501/2e/com.apple.documentVersions
bash-3.2# xattr -xl *
com.apple.genstore.info:
00000000  62 70 6C 69 73 74 30 30 D1 01 02 5E 4E 53 44 6F  |bplist00...^NSDo|
00000010  63 75 6D 65 6E 74 49 6E 66 6F D1 03 04 5F 10 14  |cumentInfo..._..|
00000020  4E 53 50 72 65 73 65 72 76 61 74 69 6F 6E 52 65  |NSPreservationRe|
00000030  61 73 6F 6E 10 14 08 0B 1A 1D 34 00 00 00 00 00  |ason......4.....|
00000040  00 01 01 00 00 00 00 00 00 00 05 00 00 00 00 00  |................|
00000050  00 00 00 00 00 00 00 00 00 00 36                 |..........6|
0000005b
com.apple.genstore.orig_perms_v1:
00000000  04                                               |.|
00000001
com.apple.genstore.origdisplayname:
00000000  49 4D 47 5F 30 31 39 30 2E 6A 70 67              |IMG_0190.jpg|
0000000c
```



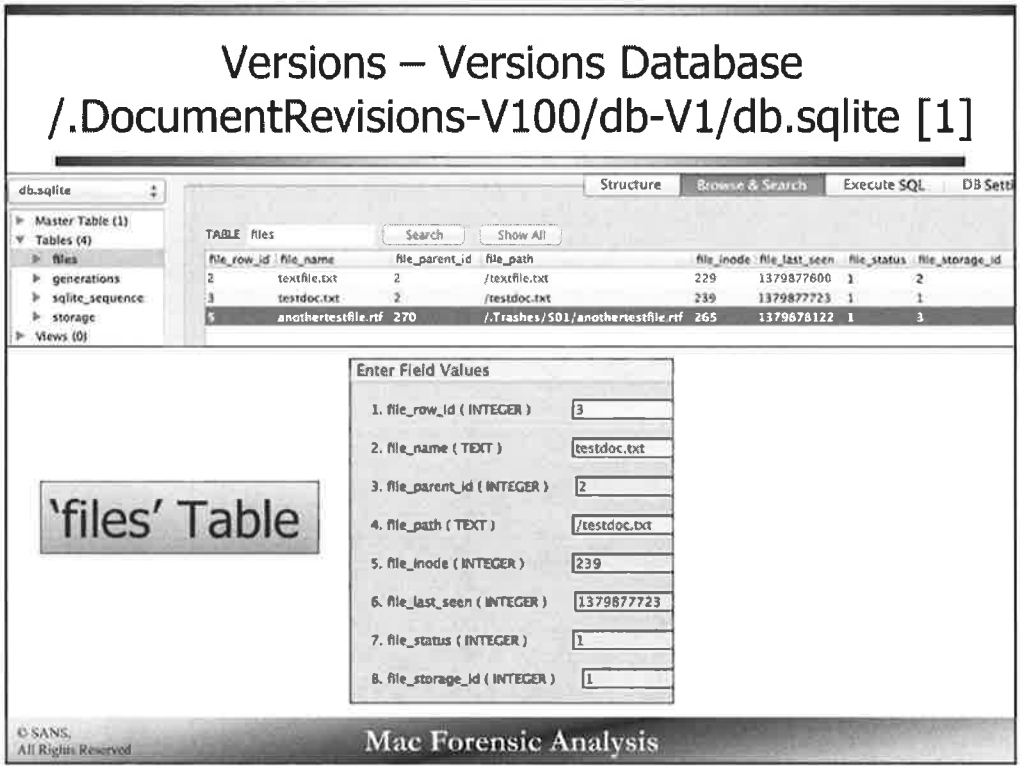
Using the command below, we can extract this binary plist (make sure you keep those dashes in there):

```
xattr -p com.apple.genstore.info <filename> | xxd -r -p | plutil -convert
xml1 - -o -
```

Once the binary property list is extracted from the `com.apple.genstore.info` attribute we can gather more metadata about the document "generation". This property list contains the keys listed above in the slide.

The `NSPreservationReason` has a number associated with why this "generation" was created. More testing and research will have to be performed to distinguish what action is required for each of these identifiers. The `NSDocumentPreviousSavedDate` contains the date the generation was previously saved.

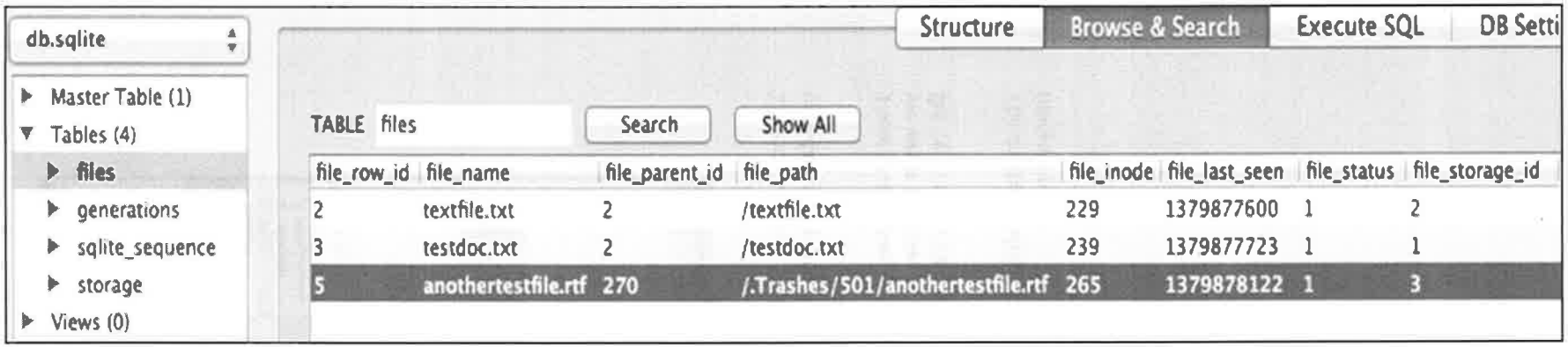
Two iCloud related keys may be stored showing when a document was last modified (`com.apple.ubiquity.moddate`) on a specific iCloud "peer" (`com.apple.ubiquity.peername`).

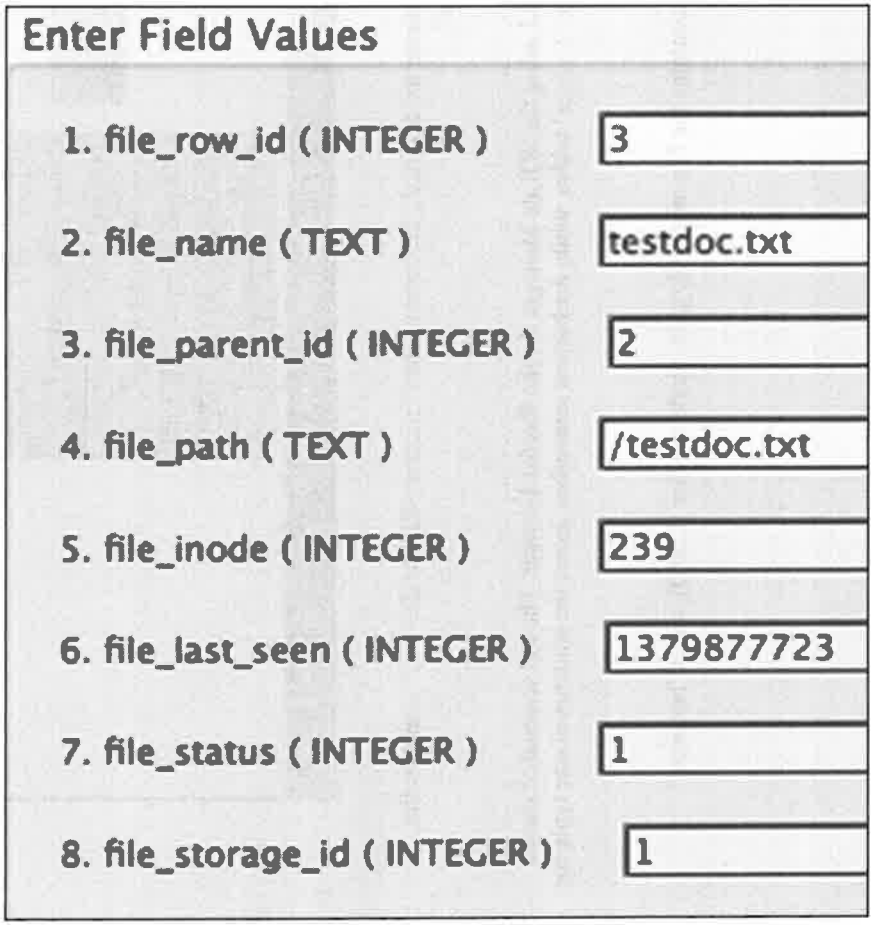


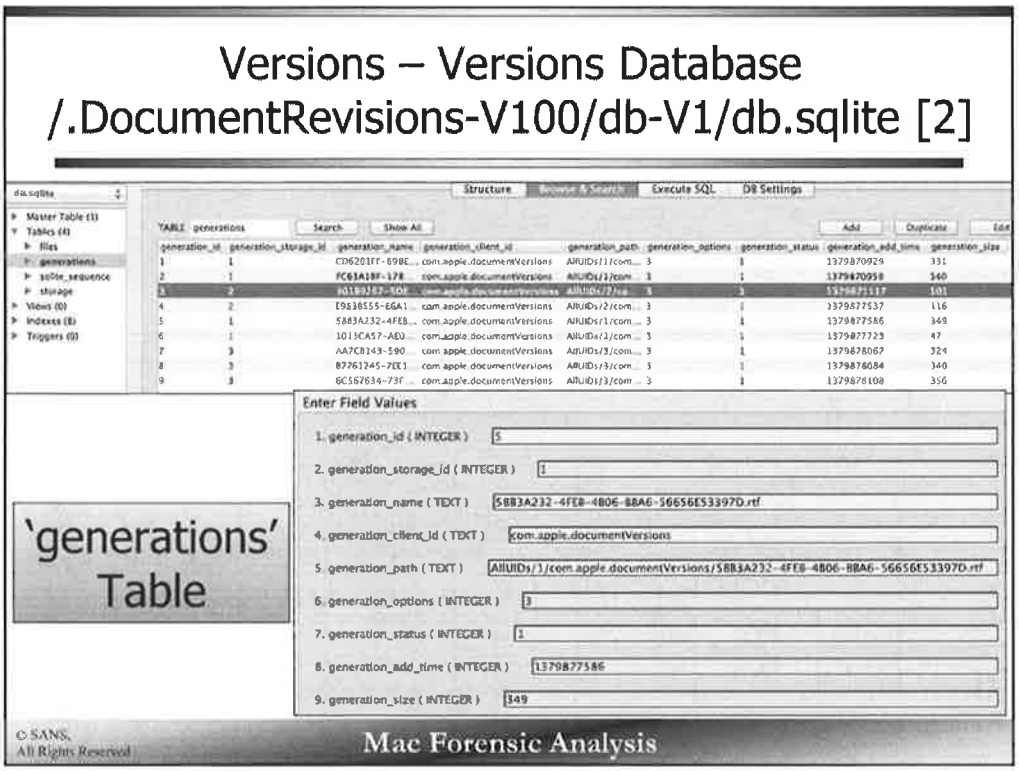
The `db.sqlite` SQLite database located in `/.DocumentRevisions-V100/db-V1/` contains the metadata for the file "versions".

These screenshots were created using the SQLite Manager for the Firefox browser. The top screenshot shows the larger database view of the '`files`' tables, while the bottom screenshot shows the contents of one tuple for the `testdoc.txt` file.

The database table, '`files`' contains the file name, file path, inode number (CNID), and a "last seen" timestamp.

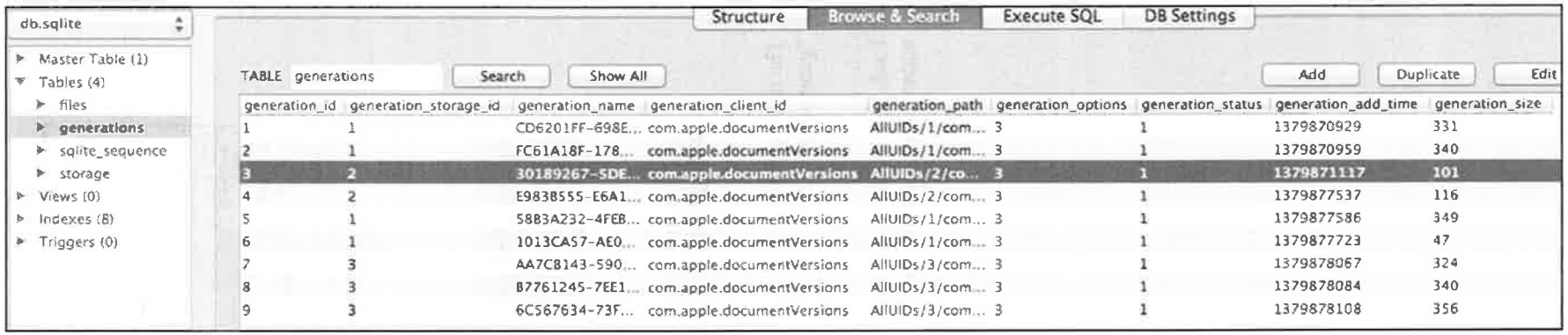

db.sqlite
Master Table (1)
Tables (4)
files
generations
sqlite_sequence
storage
Views (0)
Structure
Browse & Search
Execute SQL
DB Setti
TABLE files
Search
Show All
file_row_id file_name file_parent_id file_path file_inode file_last_seen file_status file_storage_id
2 textfile.txt 2 /textfile.txt 229 1379877600 1 2
3 testdoc.txt 2 /testdoc.txt 239 1379877723 1 1
5 anothertestfile.rtf 270 /.Trashes/501/anothertestfile.rtf 265 1379878122 1 3


Enter Field Values
1. file_row_id (INTEGER) 3
2. file_name (TEXT) testdoc.txt
3. file_parent_id (INTEGER) 2
4. file_path (TEXT) /testdoc.txt
5. file_inode (INTEGER) 239
6. file_last_seen (INTEGER) 1379877723
7. file_status (INTEGER) 1
8. file_storage_id (INTEGER) 1



The database table, 'generations' in the db.sqlite database contains the details about each document "generation".

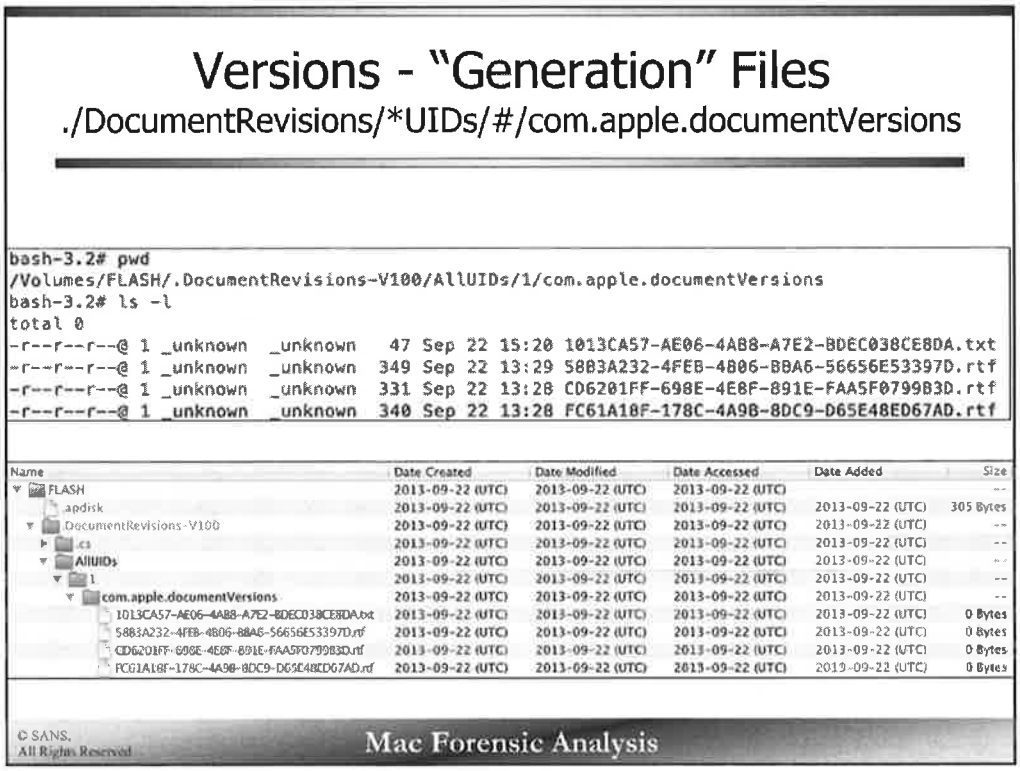
Each "generation" tuple contains the generation GUID, path to the "generation", a timestamp when it was added, and the size of the "generation".



generation_id	generation_storage_id	generation_name	generation_client_id	generation_path	generation_options	generation_status	generation_add_time	generation_size
1	1	CD6201FF-698E...	com.apple.documentVersions	AllUIDs/1/com...	3	1	1379870929	331
2	1	FC61A18F-178...	com.apple.documentVersions	AllUIDs/1/com...	3	1	1379870959	340
3	2	30189267-5DE...	com.apple.documentVersions	AllUIDs/2/co...	3	1	1379871117	101
4	2	E9838555-E6A1...	com.apple.documentVersions	AllUIDs/2/com...	3	1	1379877537	116
5	1	58B3A232-4FEB...	com.apple.documentVersions	AllUIDs/1/com...	3	1	1379877586	349
6	1	1013CA57-AE0...	com.apple.documentVersions	AllUIDs/1/com...	3	1	1379877723	47
7	3	AA7C8143-590...	com.apple.documentVersions	AllUIDs/3/com...	3	1	1379878067	324
8	3	B7761245-7EE1...	com.apple.documentVersions	AllUIDs/3/com...	3	1	1379878084	340
9	3	6C567634-73F...	com.apple.documentVersions	AllUIDs/3/com...	3	1	1379878108	356

Enter Field Values

1. generation_id (INTEGER) 5
2. generation_storage_id (INTEGER) 1
3. generation_name (TEXT) 58B3A232-4FEB-4B06-BBA6-56656E53397D.rtf
4. generation_client_id (TEXT) com.apple.documentVersions
5. generation_path (TEXT) AllUIDs/1/com.apple.documentVersions/58B3A232-4FEB-4B06-BBA6-56656E53397D.rtf
6. generation_options (INTEGER) 3
7. generation_status (INTEGER) 1
8. generation_add_time (INTEGER) 1379877586
9. generation_size (INTEGER) 349



In the top screenshot, the output of an `ls` command in Terminal shows the document "generations" for one file.

The fifth column in this Terminal output is the expected file size for each "generation" of the file. It may look like these files are all individual files, but the OS X system is only supposed to store one copy of each document, right?

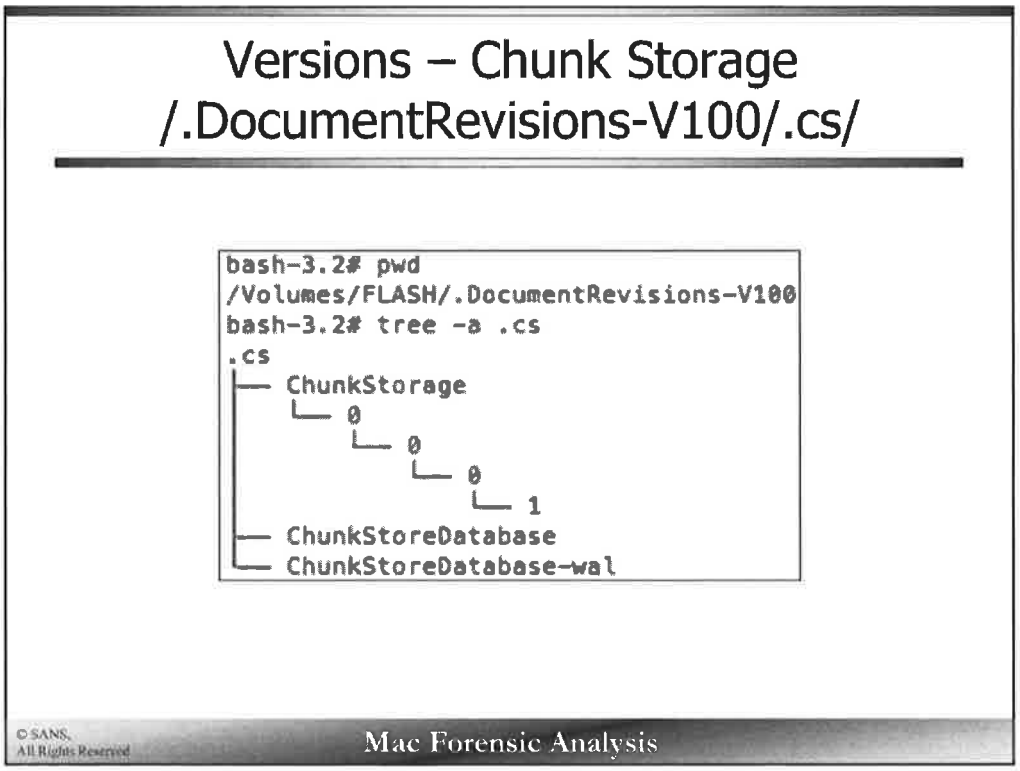
The bottom screenshot shows the same files in BlackLight. Note the sizes are 0 bytes…so where is the data actually stored? ..ChunkStorage!

On a side note, notice how the file extension for these files changes from `.rtf` to `.txt` over a period of time. This was caused by the user selecting "`Make Plaintext`" from the Format menu in TextEdit.

```
bash-3.2# pwd
/Volumes/FLASH/.DocumentRevisions-V100/AllUIDs/1/com.apple.documentVersions
bash-3.2# ls -l
total 0
-r--r--r--@ 1 _unknown  _unknown   47 Sep 22 15:20 1013CA57-AE06-4AB8-A7E2-BDEC038CE8DA.txt
-r--r--r--@ 1 _unknown  _unknown  349 Sep 22 13:29 58B3A232-4FEB-4B06-BBA6-56656E53397D.rtf
-r--r--r--@ 1 _unknown  _unknown  331 Sep 22 13:28 CD6201FF-698E-4E8F-891E-FAA5F0799B3D.rtf
-r--r--r--@ 1 _unknown  _unknown  340 Sep 22 13:28 FC61A18F-178C-4A9B-8DC9-D65E48ED67AD.rtf
```

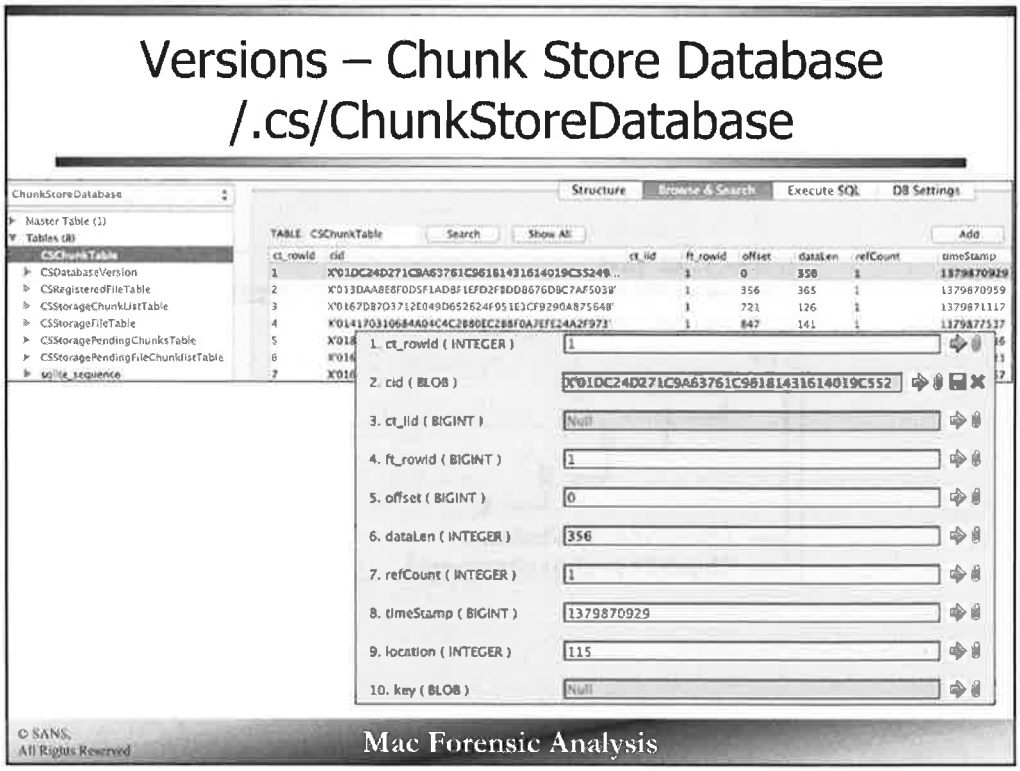
188

Name	Date Created	Date Modified	Date Accessed	Date Added	Size
FLASH	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)		--
.apdisk	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	305 Bytes
.DocumentRevisions-V100	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	--
.cs	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	--
AllUIDs	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	--
1	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	--
com.apple.documentVersions	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	--
1013CA57-AE06-4AB8-A7E2-BDEC038CE8DA.txt	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	0 Bytes
58B3A232-4FEB-4B06-BBA6-56656E53397D.rtf	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	0 Bytes
CD6201FF-698E-4E8F-891E-FAA5F0799B3D.rtf	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	0 Bytes
FC61A18F-178C-4A9B-8DC9-D65E48ED67AD.rtf	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	2013-09-22 (UTC)	0 Bytes



The data is saved in something called Chunk Storage. All the expected data supposedly stored in the "generations" files are stored in a file as "chunks" that is managed by a SQLite database.

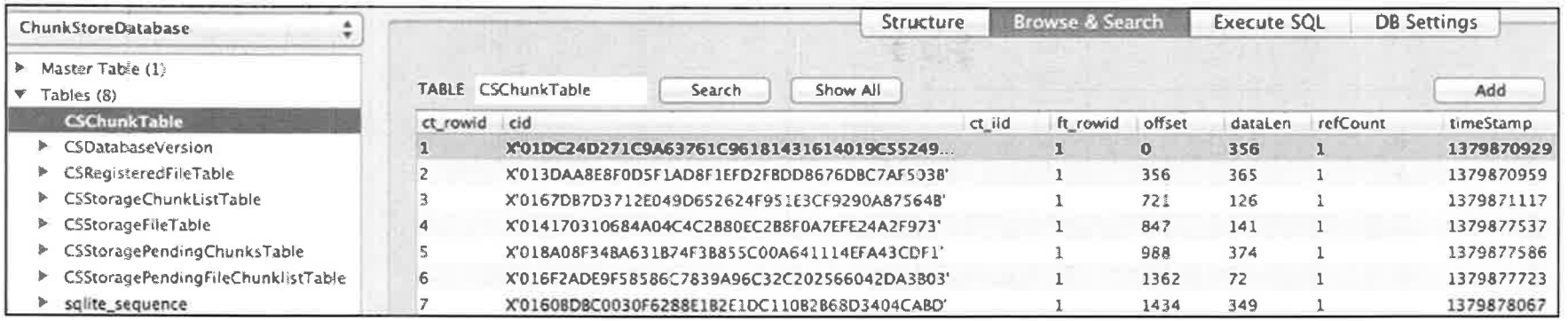
The screenshot above shows a tree output showing the structure of the hidden `.cs` directory located in the `/.DocumentRevisions-V100/` directory. The Chunk Storage files are saved in the nested sub-directories under the `ChunkStorage` directory. The SQLite database controlling this storage is the `ChunkStoreDatabase` file.

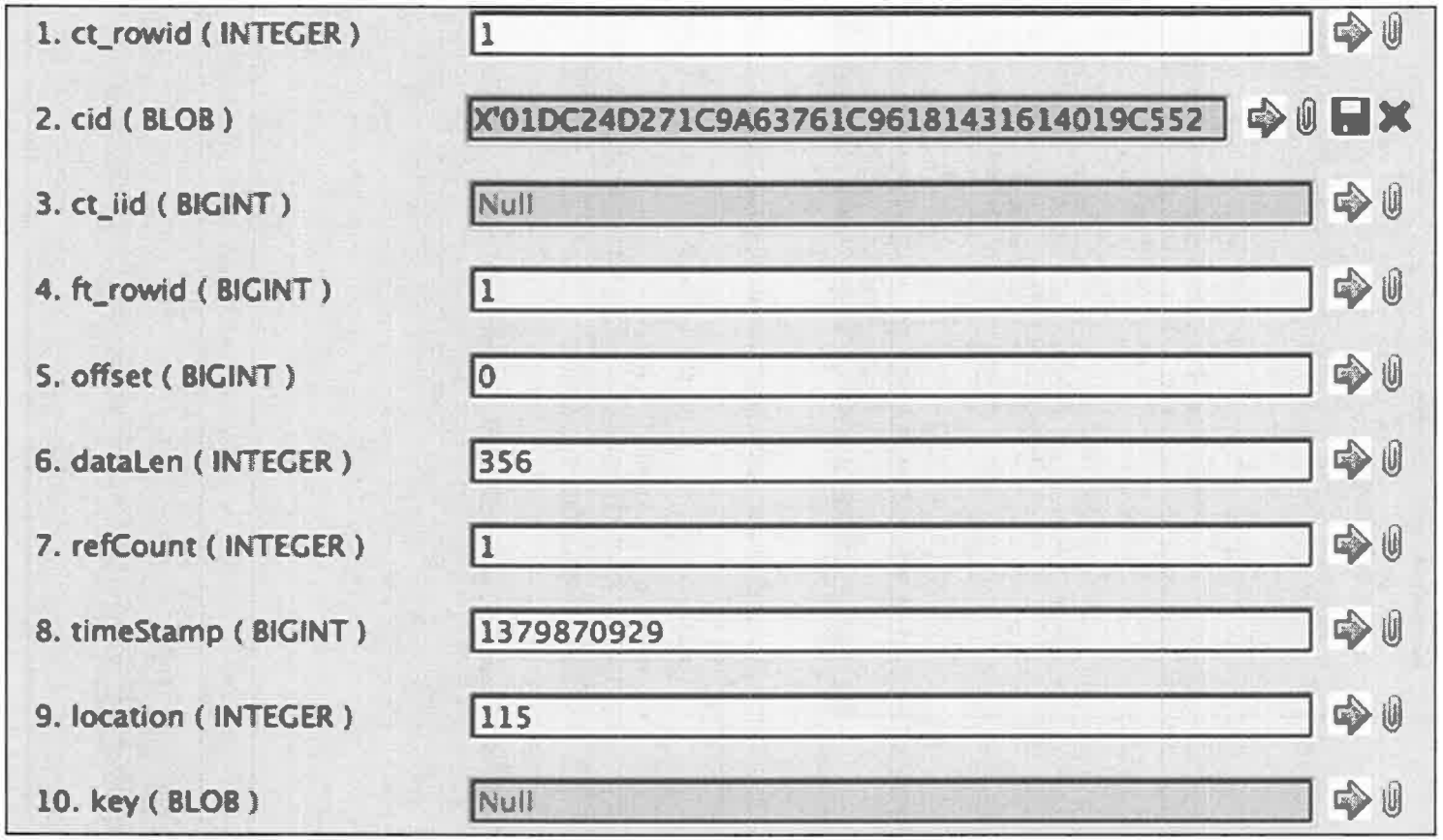


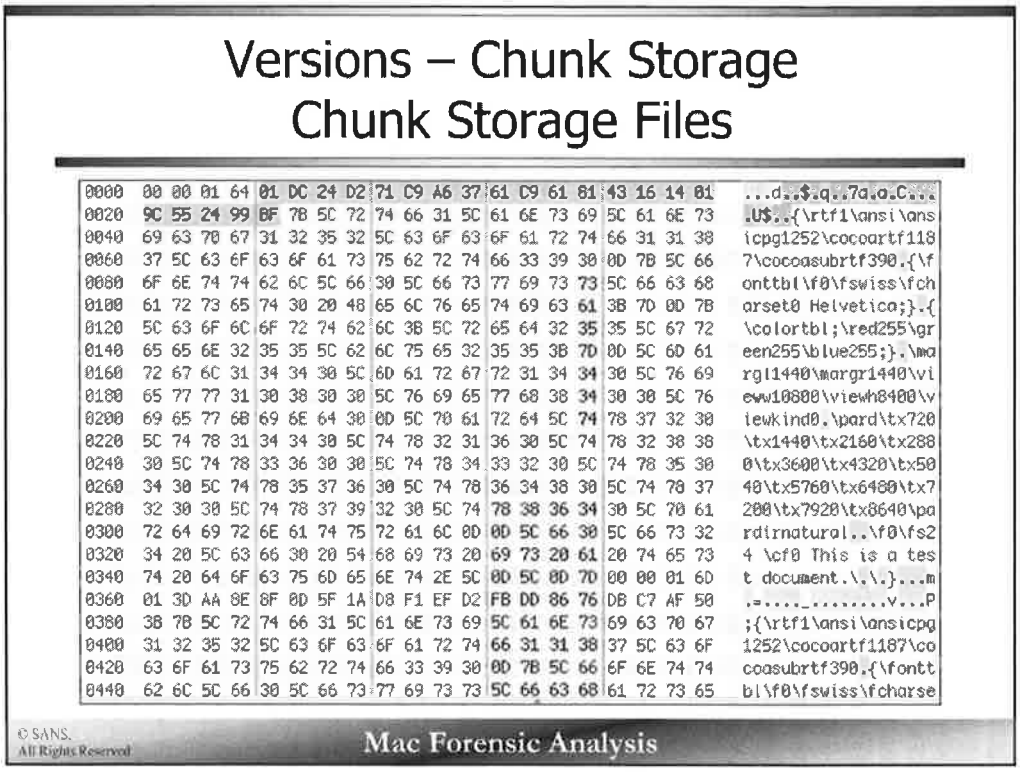
The SQLite database, `ChunkStoreDatabase`, contains the information needed to get at the stored "chunks" and to reassemble them as needed.

The `CSChunkTable`, shown in the top screenshot contains a row for each "chunk". Each "chunk" is defined by the following:

- `CID` – Chunk ID
- `Offset` – Offset in the ChuckStorage data file
- `dataLen` - a data length
- `timeStamp` – When this "chunk" was stored


ChunkStoreDatabase
Master Table (1)
Tables (8)
CSChunkTable
CSDatabaseVersion
CSRegisteredFileTable
CSStorageChunkListTable
CSStorageFileTable
CSStoragePendingChunksTable
CSStoragePendingFileChunklistTable
sqlite_sequence
Structure
Browse & Search
Execute SQL
DB Settings
TABLE CSChunkTable
Search
Show All
Add
ct_rowid | cid | ct_iid | ft_rowid | offset | dataLen | refCount | timeStamp
1 | X'01DC24D271C9A63761C96181431614019C55249... | | 1 | 0 | 356 | 1 | 1379870929
2 | X'013DAA8E8F0D5F1AD8F1EFD2FBDD8676DBC7AF5038' | | 1 | 356 | 365 | 1 | 1379870959
3 | X'0167DB7D3712E049D652624F951E3CF9290A87564B' | | 1 | 721 | 126 | 1 | 1379871117
4 | X'014170310684A04C4C2B80EC2B8F0A7EFE24A2F973' | | 1 | 847 | 141 | 1 | 1379877537
5 | X'018A08F348A631B74F3B855C00A641114EFA43CDF1' | | 1 | 988 | 374 | 1 | 1379877586
6 | X'016F2ADE9F58586C7839A96C32C202566042DA3B03' | | 1 | 1362 | 72 | 1 | 1379877723
7 | X'01608D8C0030F6288E1B2E1DC11082B68D3404CABD' | | 1 | 1434 | 349 | 1 | 1379878067

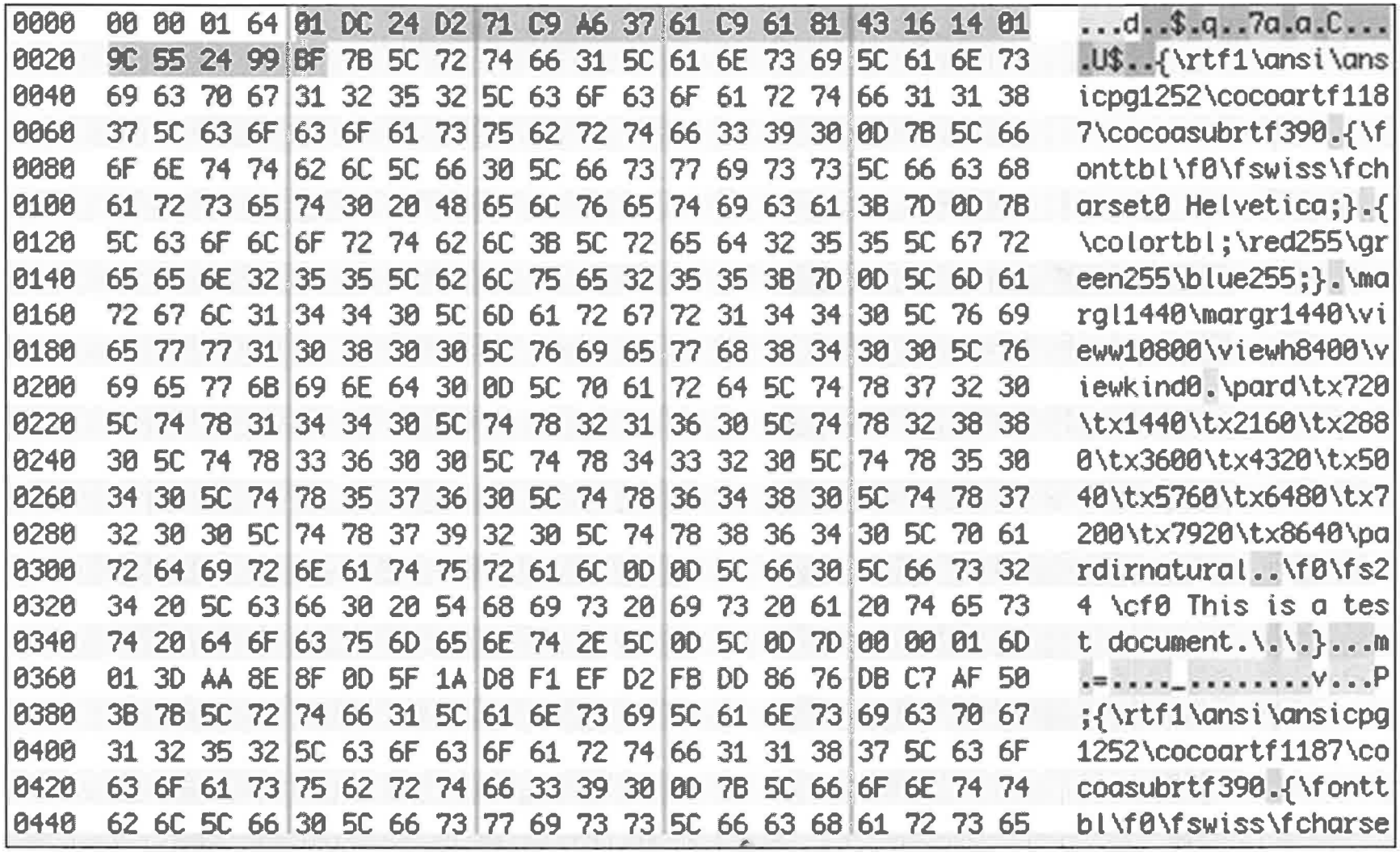

1. ct_rowid (INTEGER)
1
2. cid (BLOB)
X'01DC24D271C9A63761C96181431614019C552
3. ct_iid (BIGINT)
Null
4. ft_rowid (BIGINT)
1
5. offset (BIGINT)
0
6. dataLen (INTEGER)
356
7. refCount (INTEGER)
1
8. timeStamp (BIGINT)
1379870929
9. location (INTEGER)
115
10. key (BLOB)
Null



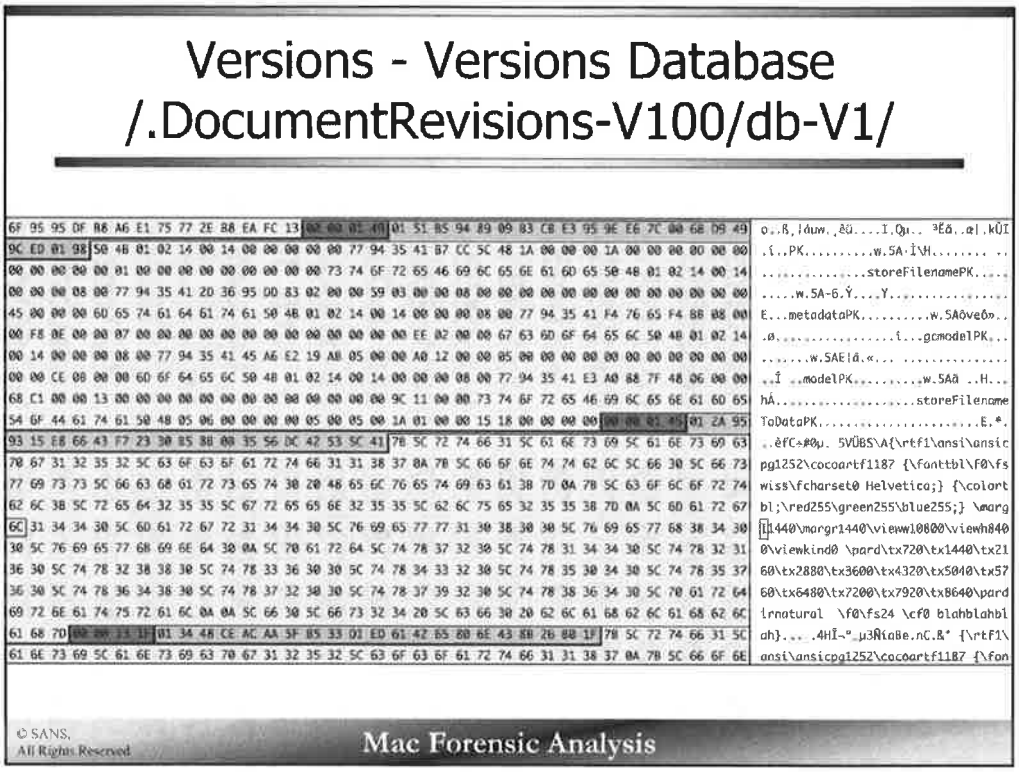
Each Chunk Storage data file located in the nested sub-directories of the `/.DocumentRevisions/.cs/` have a structure:

- First four bytes – Size of the chuck record – `0x00000164` (356)
- Next 21 bytes – Chunk ID (CID) – `0x01DC24D271C9A63761C96181431614019C552499BF`
- Rest of bytes – Chunk Contents

To find and parse the rest of the chunks, repeat the process.



```
0000  00 00 01 64 01 DC 24 D2 71 C9 A6 37 61 C9 61 81 43 16 14 01  ...d..$.q..7a.a.C...
0020  9C 55 24 99 BF 7B 5C 72 74 66 31 5C 61 6E 73 69 5C 61 6E 73  .U$..{\rtf1\ansi\ans
0040  69 63 70 67 31 32 35 32 5C 63 6F 63 6F 61 72 74 66 31 31 38  icpg1252\cocoartf118
0060  37 5C 63 6F 63 6F 61 73 75 62 72 74 66 33 39 30 0D 7B 5C 66  7\cocoasubrtf390.{\f
0080  6F 6E 74 74 62 6C 5C 66 30 5C 66 73 77 69 73 73 5C 66 63 68  onttbl\f0\fswiss\fch
0100  61 72 73 65 74 30 20 48 65 6C 76 65 74 69 63 61 3B 7D 0D 7B  arset0 Helvetica;}.{
0120  5C 63 6F 6C 6F 72 74 62 6C 3B 5C 72 65 64 32 35 35 5C 67 72  \colortbl;\red255\gr
0140  65 65 6E 32 35 35 5C 62 6C 75 65 32 35 35 3B 7D 0D 5C 6D 61  een255\blue255;}.\ma
0160  72 67 6C 31 34 34 30 5C 6D 61 72 67 72 31 34 34 30 5C 76 69  rgl1440\margr1440\vi
0180  65 77 77 31 30 38 30 30 5C 76 69 65 77 68 38 34 30 30 5C 76  eww10800\viewh8400\v
0200  69 65 77 6B 69 6E 64 30 0D 5C 70 61 72 64 5C 74 78 37 32 30  iewkind0.\pard\tx720
0220  5C 74 78 31 34 34 30 5C 74 78 32 31 36 30 5C 74 78 32 38 38  \tx1440\tx2160\tx288
0240  30 5C 74 78 33 36 30 30 5C 74 78 34 33 32 30 5C 74 78 35 30  0\tx3600\tx4320\tx50
0260  34 30 5C 74 78 35 37 36 30 5C 74 78 36 34 38 30 5C 74 78 37  40\tx5760\tx6480\tx7
0280  32 30 30 5C 74 78 37 39 32 30 5C 74 78 38 36 34 30 5C 70 61  200\tx7920\tx8640\pa
0300  72 64 69 72 6E 61 74 75 72 61 6C 0D 0D 5C 66 30 5C 66 73 32  rdirnatural..\f0\fs2
0320  34 20 5C 63 66 30 20 54 68 69 73 20 69 73 20 61 20 74 65 73  4 \cf0 This is a tes
0340  74 20 64 6F 63 75 6D 65 6E 74 2E 5C 0D 5C 0D 7D 00 00 01 6D  t document.\.\.}...m
0360  01 3D AA 8E 8F 0D 5F 1A D8 F1 EF D2 FB DD 86 76 DB C7 AF 50  .=...._........v...P
0380  3B 7B 5C 72 74 66 31 5C 61 6E 73 69 5C 61 6E 73 69 63 70 67  ;{\rtf1\ansi\ansicpg
0400  31 32 35 32 5C 63 6F 63 6F 61 72 74 66 31 31 38 37 5C 63 6F  1252\cocoartf1187\co
0420  63 6F 61 73 75 62 72 74 66 33 39 30 0D 7B 5C 66 6F 6E 74 74  coasubrtf390.{\fontt
0440  62 6C 5C 66 30 5C 66 73 77 69 73 73 5C 66 63 68 61 72 73 65  bl\f0\fswiss\fcharse
```



Another example of the Chunk Storage data file shown in Synalyze It!. This example shows three records:

1st Record –

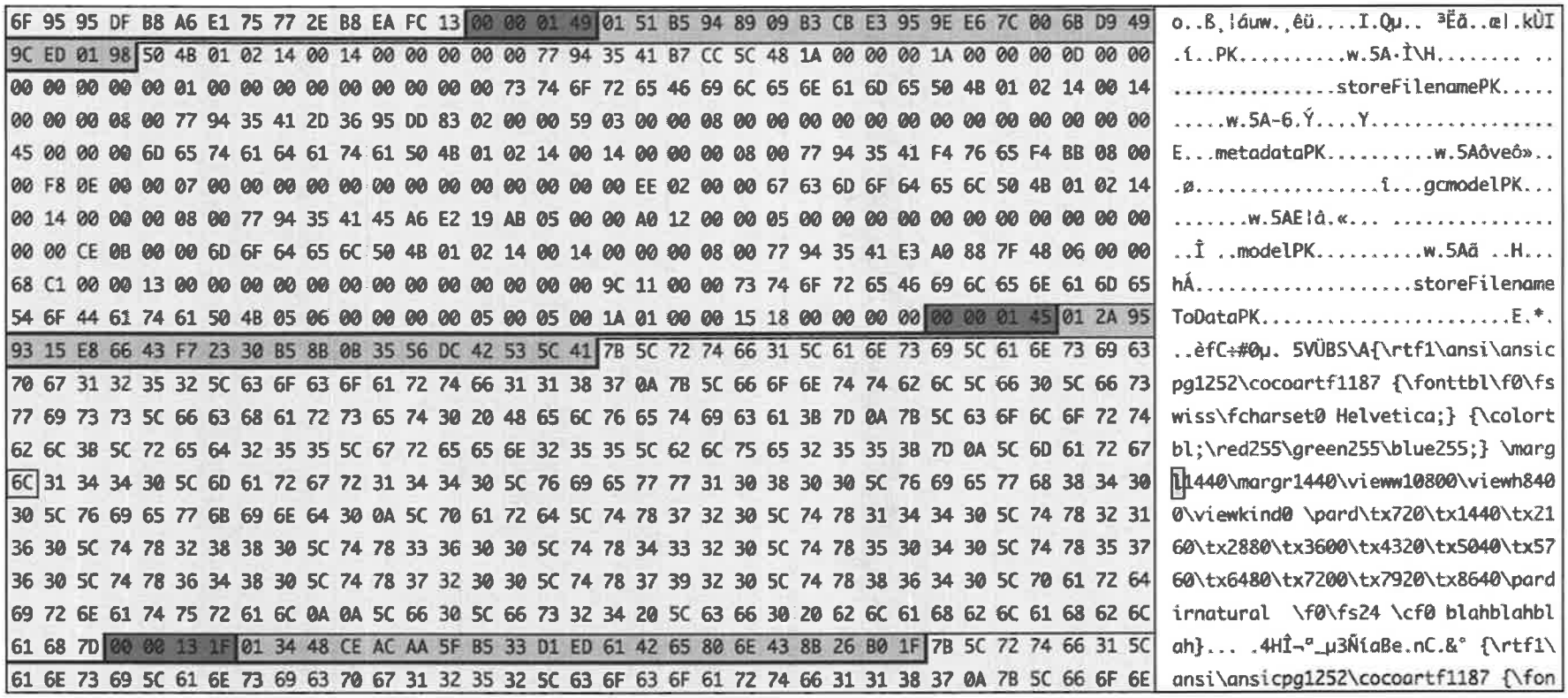
- First four bytes – Size of the chuck record – `0x00000149` (329)
- Next 21 bytes – Chunk ID (CID) – `0x0151B5948909B3CBE3959EE67C006BD9499CED0198`
- Rest of bytes – Chunk Contents

2nd Record –

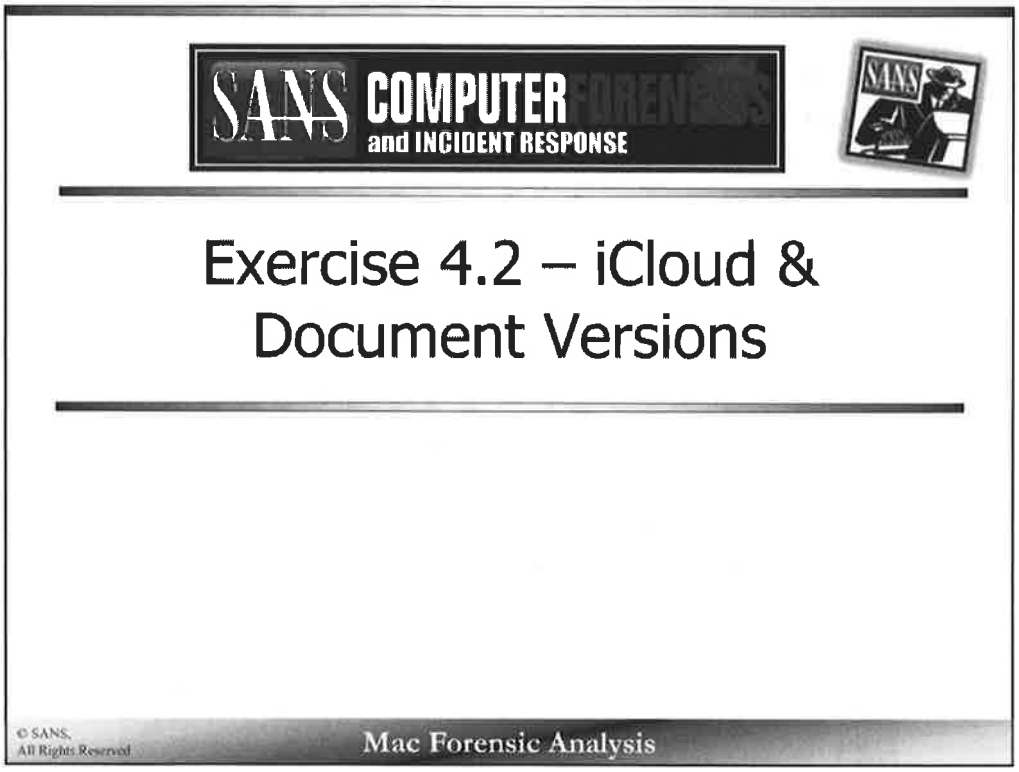
- First four bytes – Size of the chuck record – `0x00000145` (325)
- Next 21 bytes – Chunk ID (CID) – `0x012A959315E86643F72330B58B0B3556DC42535C41`
- Rest of bytes – Chunk Contents

3rd Record –

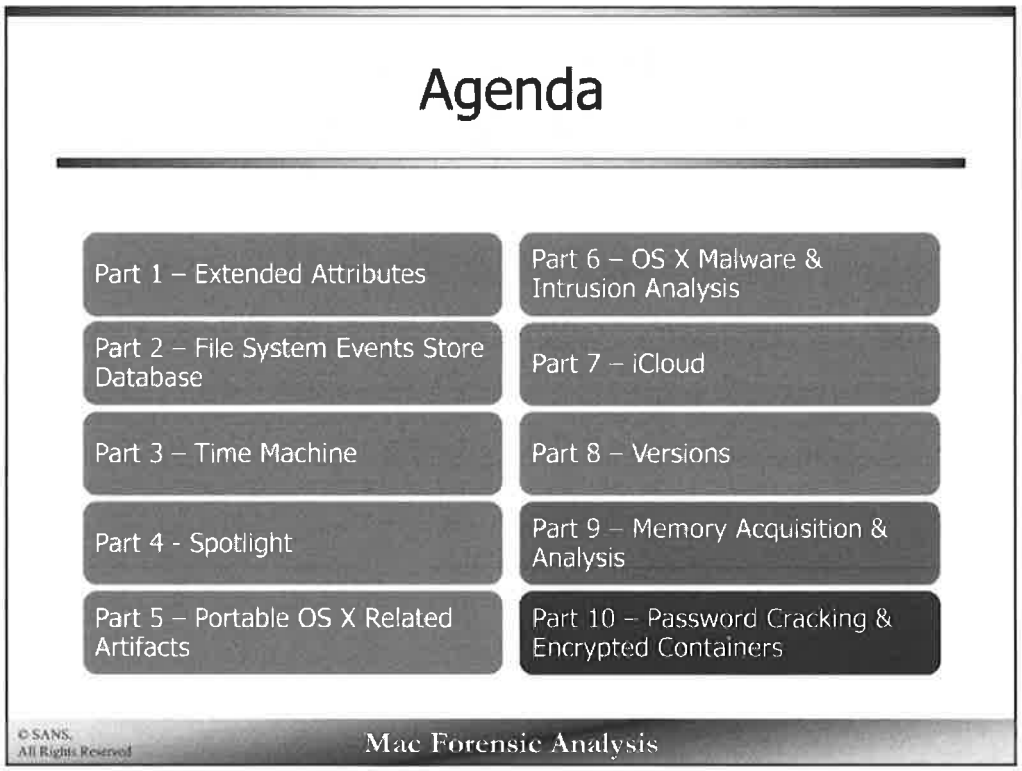
- First four bytes – Size of the chuck record – `0x0000131F` (4895)
- Next 21 bytes – Chunk ID (CID) – `0x013448CEACAA5FB533D1ED614265806E438B26B01F`
- Rest of bytes – Chunk Contents



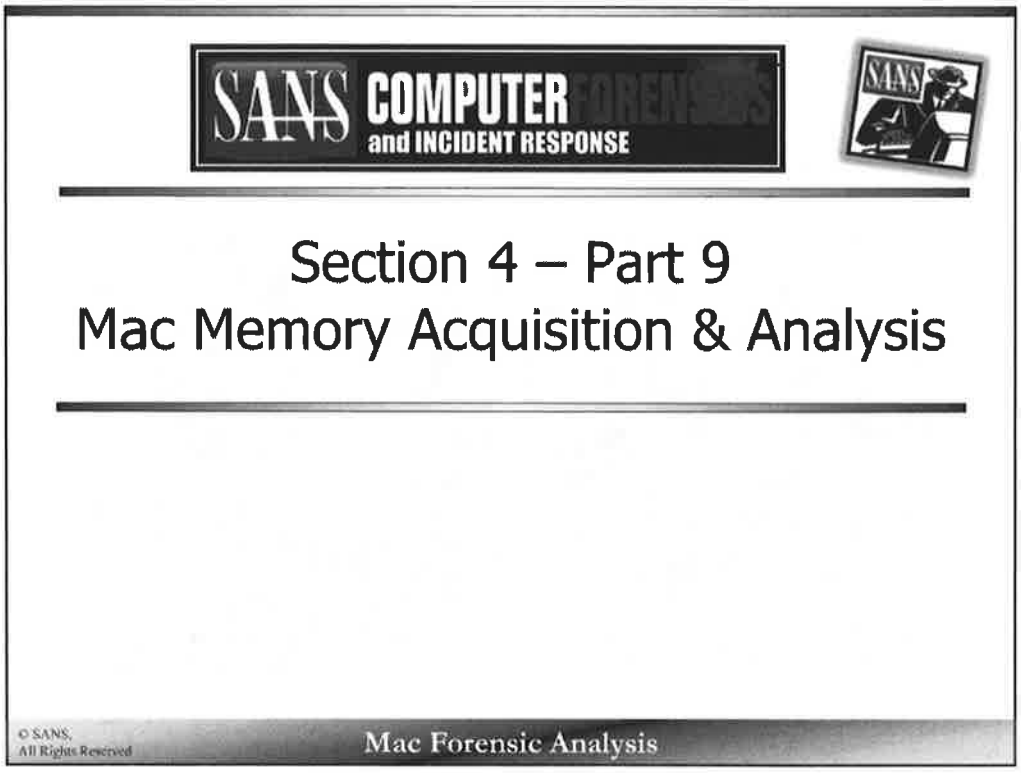
```
6F 95 95 DF B8 A6 E1 75 77 2E B8 EA FC 13 00 00 01 49 01 51 B5 94 89 09 B3 CB E3 95 9E E6 7C 00 6B D9 49   o..ß¸¦áuw.¸êü....I.Q..³Ëã..æ|.kÙI
9C ED 01 98 50 4B 01 02 14 00 14 00 00 00 00 00 77 94 35 41 B7 CC 5C 48 1A 00 00 00 1A 00 00 00 0D 00 00   .í..PK.....w.5A·Ì\H.....
00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 73 74 6F 72 65 46 69 6C 65 6E 61 6D 65 50 4B 01 02 14 00 14   .....storeFilenamePK.....
00 00 00 08 00 77 94 35 41 2D 36 95 DD 83 02 00 00 59 03 00 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00   .....w.5A-6.Ý....Y.....
45 00 00 00 6D 65 74 61 64 61 74 61 50 4B 01 02 14 00 14 00 00 00 08 00 77 94 35 41 F4 76 65 F4 BB 08 00   E...metadataPK.....w.5Aôveô»..
00 F8 0E 00 00 07 00 00 00 00 00 00 00 00 00 00 00 00 EE 02 00 00 67 63 6D 6F 64 65 6C 50 4B 01 02 14   .ø.....î...gcmodelPK...
00 14 00 00 00 08 00 77 94 35 41 45 A6 E2 19 AB 05 00 00 A0 12 00 00 05 00 00 00 00 00 00 00 00 00 00 00   .....w.5AE¦â.«...
00 00 CE 0B 00 00 6D 6F 64 65 6C 50 4B 01 02 14 00 14 00 00 00 08 00 77 94 35 41 E3 A0 88 7F 48 06 00 00   ..Î ..modelPK.....w.5Aã ..H...
68 C1 00 00 13 00 00 00 00 00 00 00 00 00 00 00 00 00 9C 11 00 00 73 74 6F 72 65 46 69 6C 65 6E 61 6D 65   hÁ.....storeFilename
54 6F 44 61 74 61 50 4B 05 06 00 00 00 00 05 00 05 00 1A 01 00 00 15 18 00 00 00 00 00 00 01 45 01 2A 95   ToDataPK.....E.*.
93 15 E8 66 43 F7 23 30 B5 8B 0B 35 56 DC 42 53 5C 41 7B 5C 72 74 66 31 5C 61 6E 73 69 5C 61 6E 73 69 63   ..èfC÷#0µ. 5VÜBS\A{\rtf1\ansi\ansic
70 67 31 32 35 32 5C 63 6F 63 6F 61 72 74 66 31 31 38 37 0A 7B 5C 66 6F 6E 74 74 62 6C 5C 66 30 5C 66 73   pg1252\cocoartf1187 {\fonttbl\f0\fs
77 69 73 73 5C 66 63 68 61 72 73 65 74 30 20 48 65 6C 76 65 74 69 63 61 3B 7D 0A 7B 5C 63 6F 6C 6F 72 74   wiss\fcharset0 Helvetica;} {\colort
62 6C 38 5C 72 65 64 32 35 35 5C 67 72 65 65 6E 32 35 35 5C 62 6C 75 65 32 35 35 3B 7D 0A 5C 6D 61 72 67   bl;\red255\green255\blue255;} \marg
6C 31 34 34 30 5C 6D 61 72 67 72 31 34 34 30 5C 76 69 65 77 77 31 30 38 30 30 5C 76 69 65 77 68 38 34 30   l1440\margr1440\vieww10800\viewh840
30 5C 76 69 65 77 6B 69 6E 64 30 0A 5C 70 61 72 64 5C 74 78 37 32 30 5C 74 78 31 34 34 30 5C 74 78 32 31   0\viewkind0 \pard\tx720\tx1440\tx21
36 30 5C 74 78 32 38 38 30 5C 74 78 33 36 30 30 5C 74 78 34 33 32 30 5C 74 78 35 30 34 30 5C 74 78 35 37   60\tx2880\tx3600\tx4320\tx5040\tx57
36 30 5C 74 78 36 34 38 30 5C 74 78 37 32 30 30 5C 74 78 37 39 32 30 5C 74 78 38 36 34 30 5C 70 61 72 64   60\tx6480\tx7200\tx7920\tx8640\pard
69 72 6E 61 74 75 72 61 6C 0A 0A 5C 66 30 5C 66 73 32 34 20 5C 63 66 30 20 62 6C 61 68 62 6C 61 68 62 6C   irnatural  \f0\fs24 \cf0 blahblahbl
61 68 7D 00 00 13 1F 01 34 48 CE AC AA 5F B5 33 D1 ED 61 42 65 80 6E 43 8B 26 B0 1F 7B 5C 72 74 66 31 5C   ah}... .4HÎ¬ª_µ3ÑíaBe.nC.&° {\rtf1\
61 6E 73 69 5C 61 6E 73 69 63 70 67 31 32 35 32 5C 63 6F 63 6F 61 72 74 66 31 31 38 37 0A 7B 5C 66 6F 6E   ansi\ansicpg1252\cocoartf1187 {\fon
```



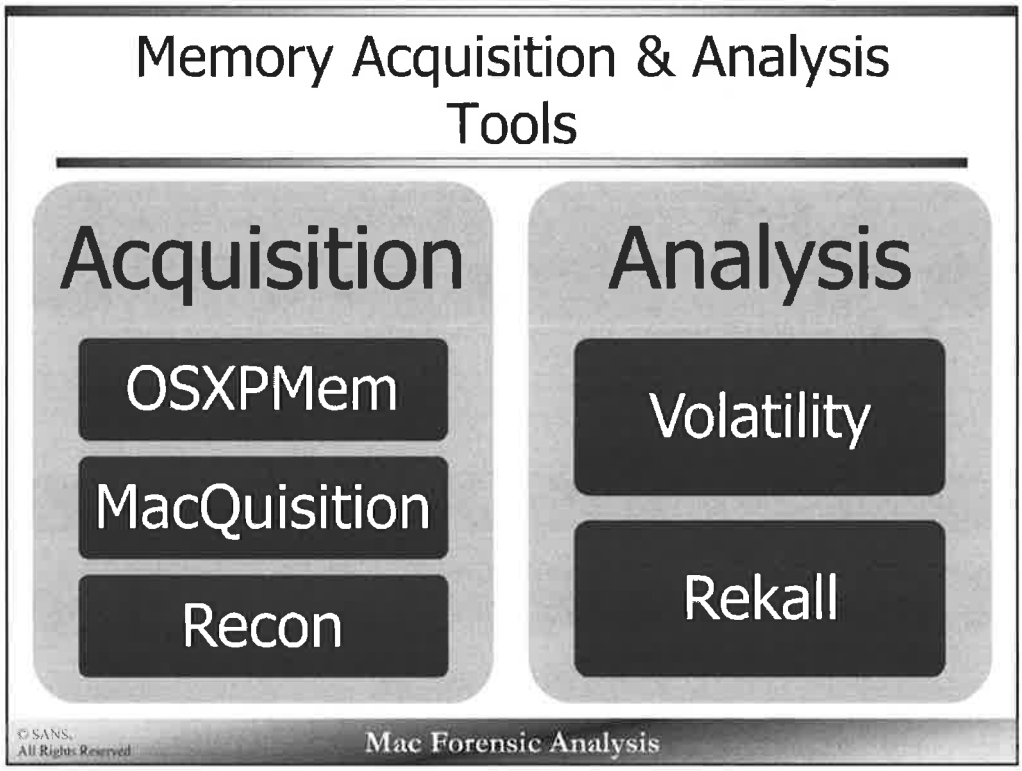
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.

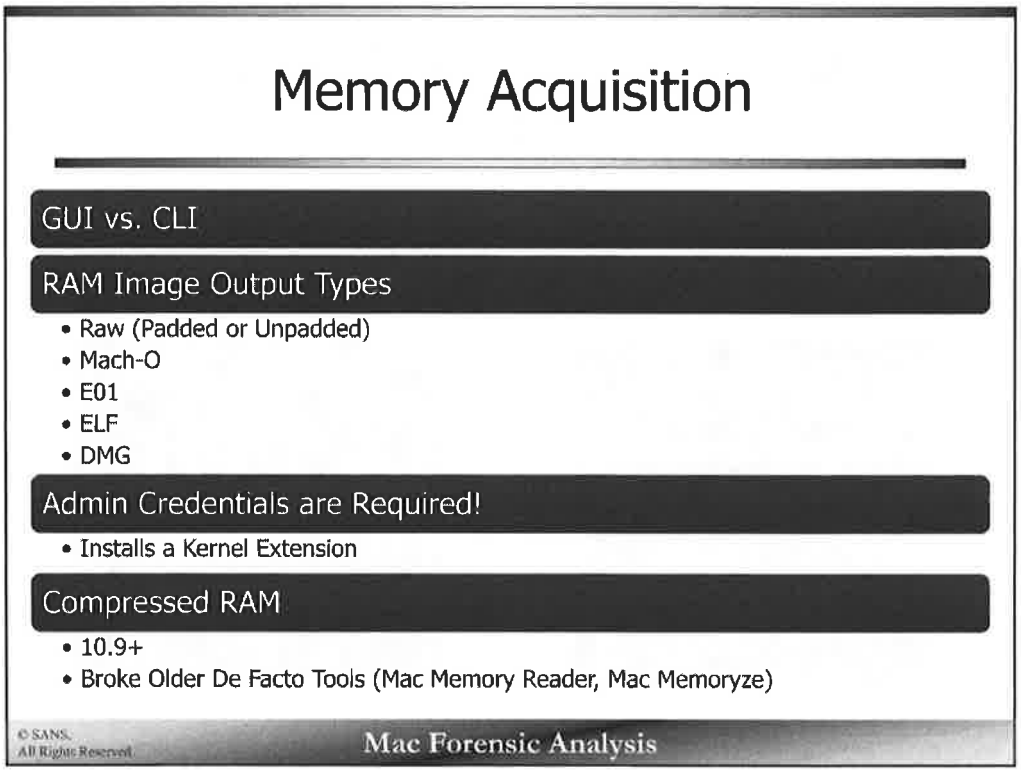


Capturing and analyzing memory from Windows based systems has been available for years, while Mac memory acquisition and analysis is relatively new.

In the past couple years or so a few developers have been able to provide us the capability to acquire RAM from Mac systems as well as the ability to analyze these RAM images. A mix of free, open source, and commercial tools are available to perform these tasks.

Each tool has their benefits and disadvantages:

- Some are free
- Some are easier to use
- Some have more capabilities
- Some are updated by the developers more frequently



Memory acquisition is the activity of capturing the system random access memory (RAM). This RAM capture contains process lists, network information, open files, and other data that may be useful in forensic analysis.

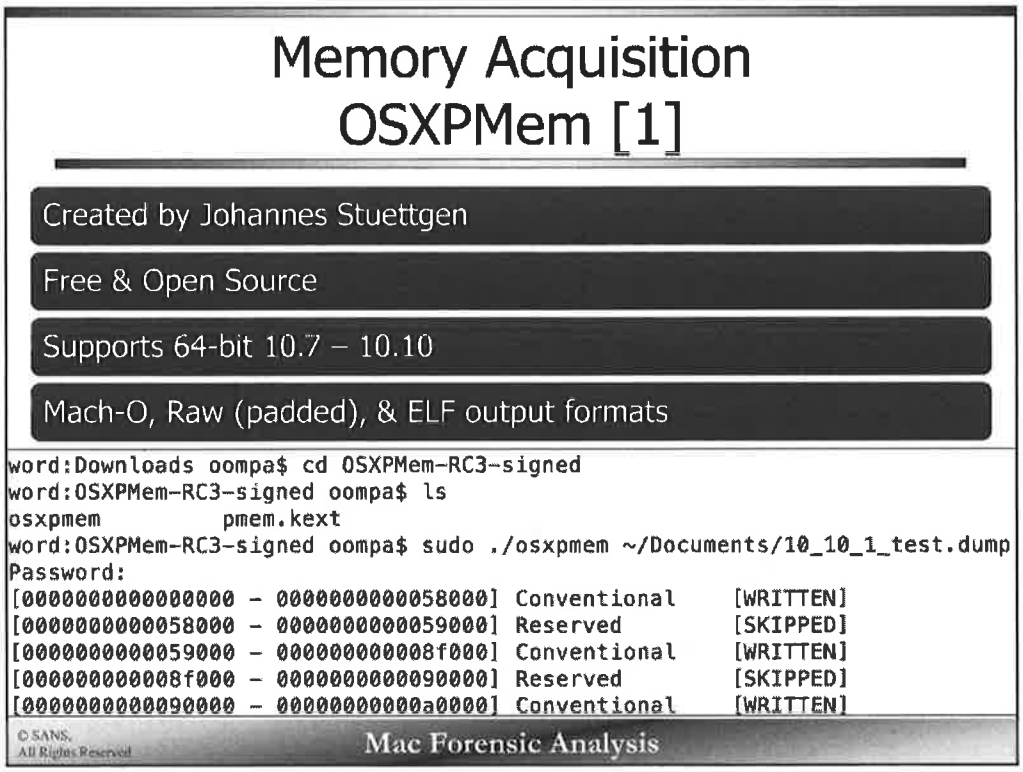
These tools have either GUI applications or command-line tools. It is worth noting that a GUI application may leave a larger memory footprint than a command line tool.

Each tool captures the RAM in their own specified formats; some tools have multiple output options while others are one and done. An analyst must be aware of the output and their preferred analysis tool. One output type may not be supported in another analysis tool. Administrator credentials are required to dump RAM, as all tools install a kernel extension.

Common output formats include:

- Raw (padded and unpadded) – DD-style format - padded format adds zeros where there are unmapped memory regions. Final RAM image size may be larger than RAM in system. Raw/Padded is one of the de-facto image standards.
- Mach-O – Mach-O file of RAM, standard format (developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/MachORuntime/Reference/reference.html). This format is another one of the de-facto image standards.
- E01 – Format used with Encase forensic software.
- ELF – Executable and Linking Format
- DMG – Disk Image format, similar to Raw and padded format.

10.9 introduced compressed RAM, this broke many of the de facto memory acquisition tools such as Mac Memory Reader and Mac Memoryze. If you want to learn more about how this works read "In Lieu of Swap: Analyzing Compressed RAM in OS X and Linux" - http://dfrws.org/2014/proceedings/DFRWS2014-1.pdf



OSXPMem is a free and open source tool created by Johannes Stuettgen. This tool is reported to support 64-bit systems from 10.7 to 10.10.

To use OSXPMem, follow these instructions:

- Download the `OSXPMem-*. tar.gz` file
- Unpack it with root privileges (use `sudo -s` or `sudo su` to get a root shell)

  `tar -xvf OSXPMem-*.tar.gz or unzip OSXPMem-*-signed.zip`
- Still in the root shell, execute the file `osxpmem` — default output is an ELF file. Use the `-f` flag with `mach` or `raw` to get different outputs formats. (if no long in the root shell use the sudo command).

  `./osxpmem memory.dump`

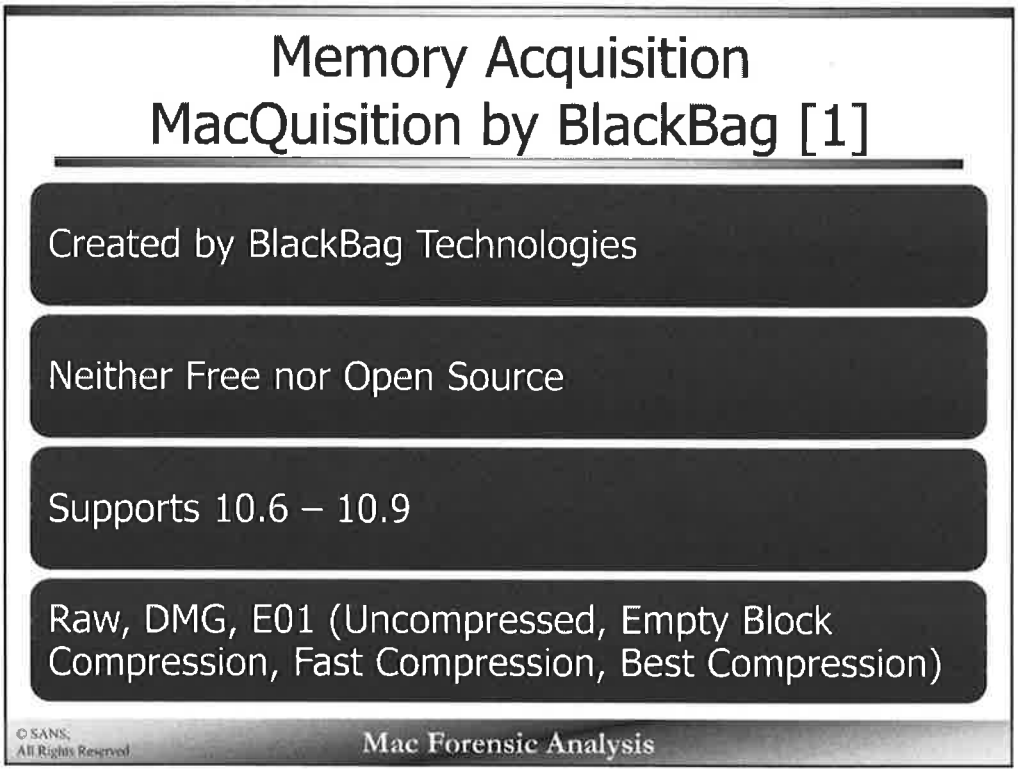
The screenshot shows the error produced if these files were not unpacked with root privileges.

```
bash-3.2# ./osxpmem /Volumes/DATA/MacMemory/osxpmem_elf.dump
Can't load kext ./pmem.kext, as it is not owned by root:wheel
dump_memory(833): Failed to load kext (Undefined error: 0)
```

References:

https://github.com/google/rekall/tree/master/tools/osx/OSXPMem

http://rekall-forensic.blogspot.com/2014/03/osx-109-memory-acquisition.html

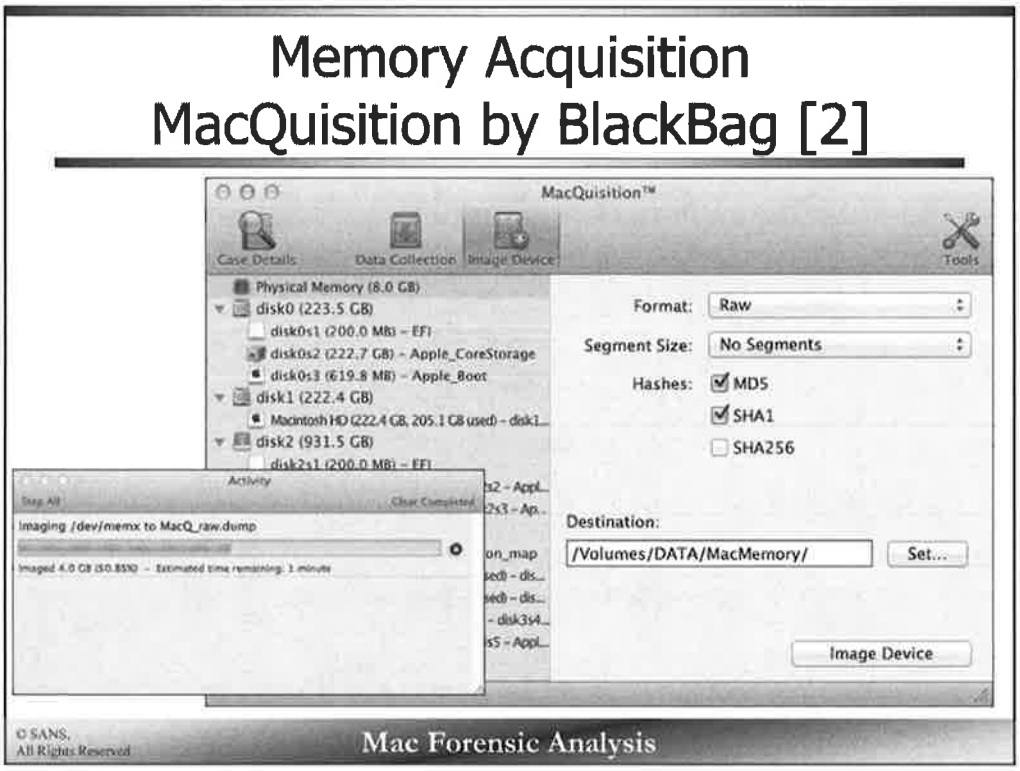


MacQuisition by BlackBag Technologies is their tool to capture all types of data in one shot. This tool allows an investigator to acquire incident response information, disk images, and RAM. While this tool makes it incredibly easy to capture everything you may need for an investigation, it does come at a price. This tool is available from `blackbagtech.com`.

This tool is only available in a GUI format run off a dongle-based thumb-drive. RAM output formats consist of Raw/Padded, DMG, and various E01 formats.

References:

blackbagtech.com/software-products/macquisition.html

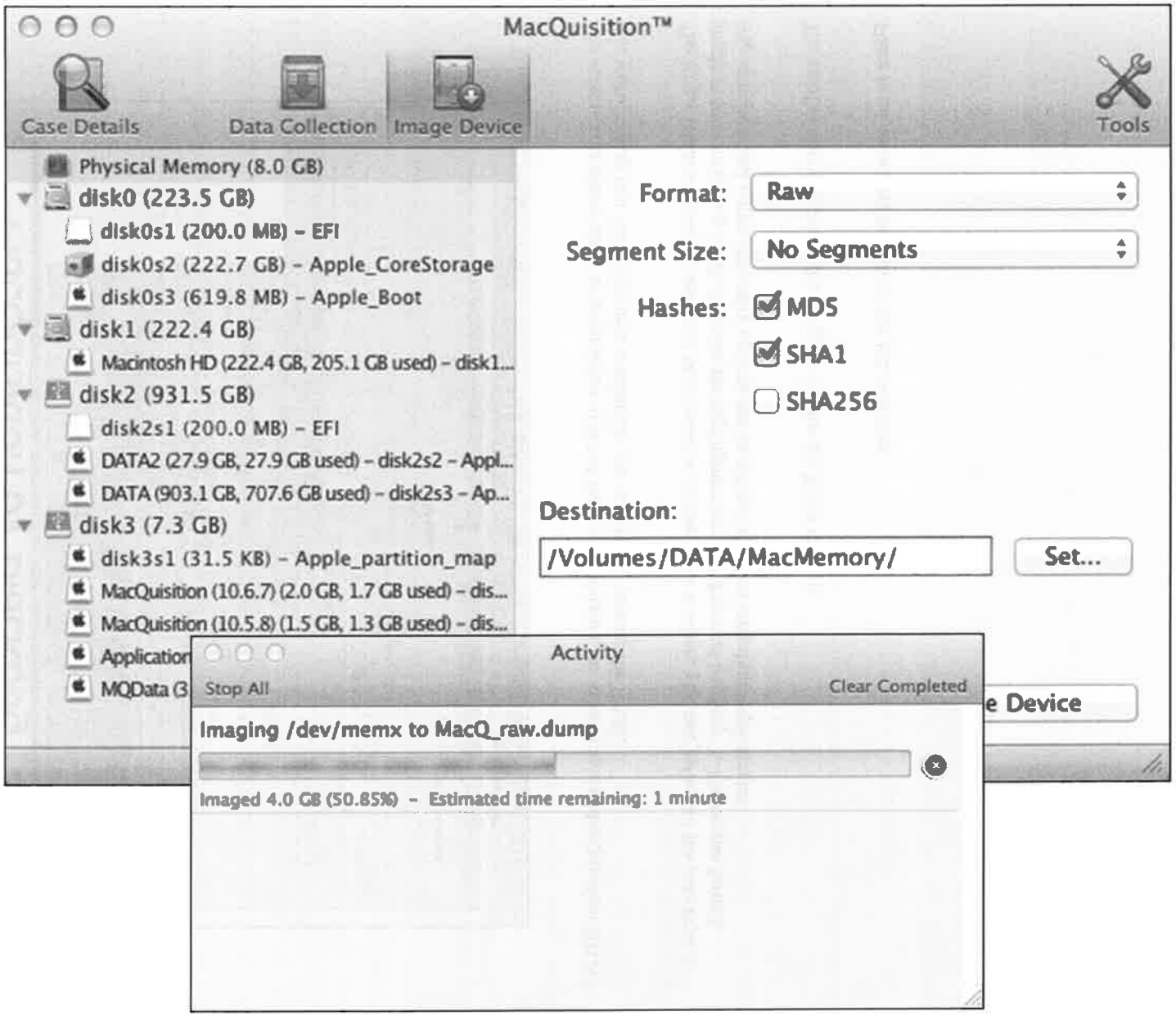


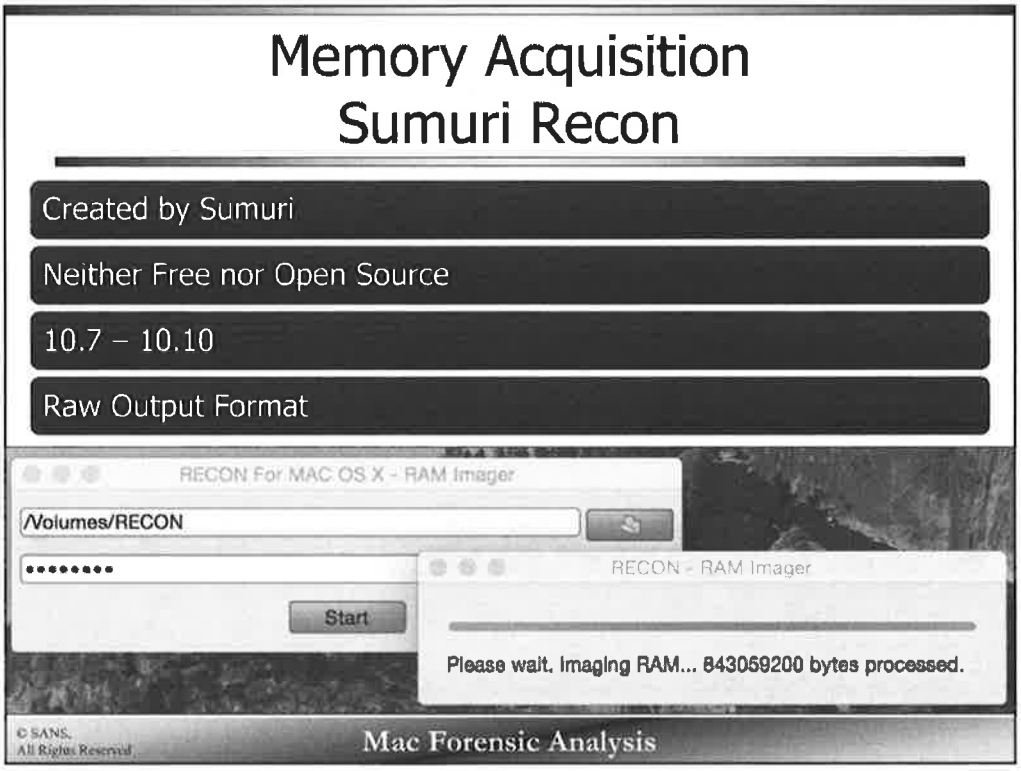
The screenshots above show an acquisition of RAM from a Mountain Lion system using MacQuisition 2012r3. It is worth noting that Administrative credentials are required to capture the RAM.

The larger window shown is the main MacQuisition screen. This is where you can gather all the data from a single system including the RAM (note the highlighted section "Physical Memory"). The output format, segmentation, and output directory options can be altered to fit investigation requirements.

The smaller window shows the progress bar for the RAM capture.

There is no command line utility for this software.


MacQuisition™
Case Details
Data Collection
Image Device
Tools
Physical Memory (8.0 GB)
disk0 (223.5 GB)
disk0s1 (200.0 MB) - EFI
disk0s2 (222.7 GB) - Apple_CoreStorage
disk0s3 (619.8 MB) - Apple_Boot
disk1 (222.4 GB)
Macintosh HD (222.4 GB, 205.1 GB used) - disk1...
disk2 (931.5 GB)
disk2s1 (200.0 MB) - EFI
DATA2 (27.9 GB, 27.9 GB used) - disk2s2 - Appl...
DATA (903.1 GB, 707.6 GB used) - disk2s3 - Ap...
disk3 (7.3 GB)
disk3s1 (31.5 KB) - Apple_partition_map
MacQuisition (10.6.7) (2.0 GB, 1.7 GB used) - dis...
MacQuisition (10.5.8) (1.5 GB, 1.3 GB used) - dis...
Format:
Raw
Segment Size:
No Segments
Hashes:
MD5
SHA1
SHA256
Destination:
/Volumes/DATA/MacMemory/
Set...
Activity
Stop All
Clear Completed
Imaging /dev/memx to MacQ_raw.dump
Imaged 4.0 GB (50.85%) - Estimated time remaining: 1 minute



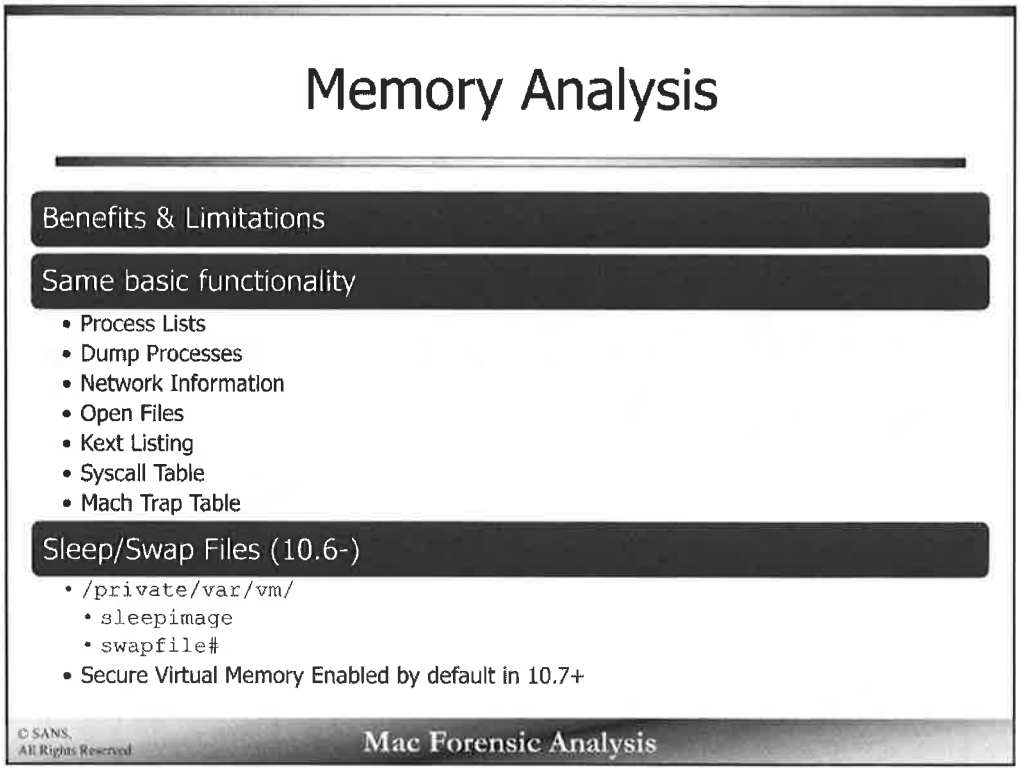
Sumuri Recon is another all encompassing suite of tools that can acquire disk images, RAM, triage analysis and live response data. It does all of this in a GUI with no command-line interface. All this functionality does come at a price and is available at `sumuri.com`.

References:

http://sumuri.com/product-category/recon/

http://sumuri.com/using-recon-to-image-ram/

http://sumuri.com/recon-adds-support-for-mac-os-x-10-10/



Memory analysis is the act of analyzing the memory to parse out data structures to determine what was happening on the system at that particular moment in time.

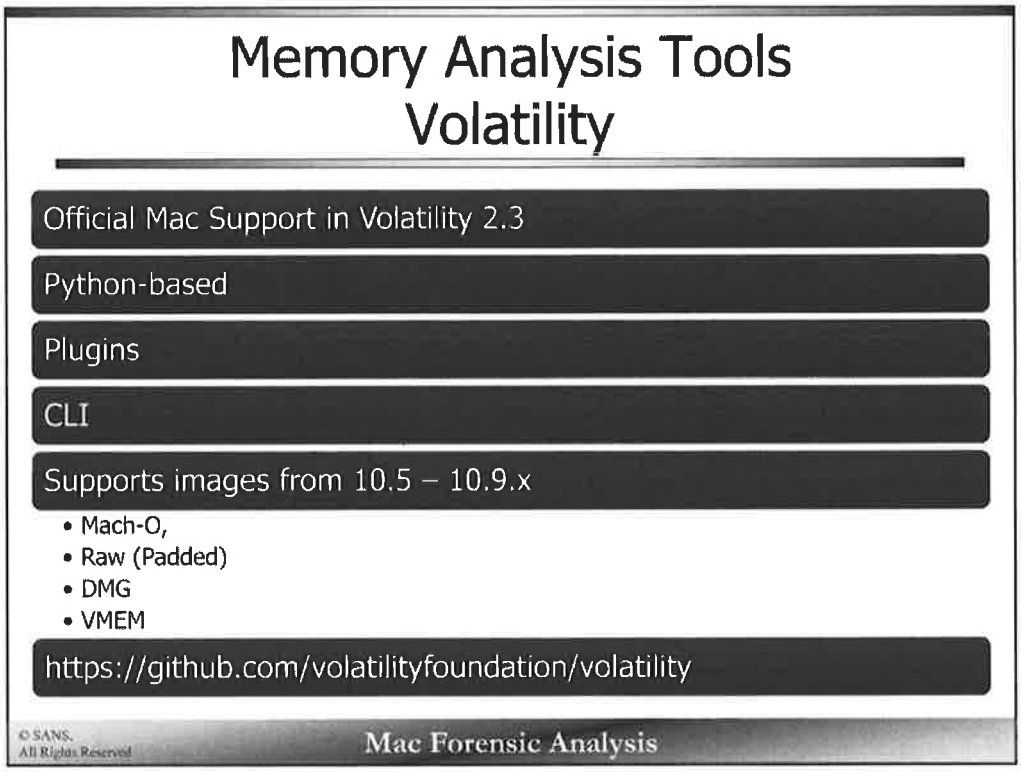
Each tool has its benefits and limitations - some tools are easier to use and some have more advanced functionality. Each tool has the same basic functionality that includes parsing the data structure to access the following information.

- Process Lists
- Dump Processes
- Network Information
- Open Files
- Kext Listing
- Syscall Table
- Mach Trap Table

Warning Caveat: Output for each image/tool combination type may vary. It doesn't hurt to perform analysis using multiple tools!

Similar to memory are hibernation and swap files. In their unencrypted formats, these files may contain passwords and other sensitive data.

The `sleepimage` file is similar with the `hiberfil.sys` file on Windows. It is created when the system goes into hibernation mode. The swap files may also contain sensitive information in their non-encrypted form. Swap files are used similarly to the Windows `pagefile.sys`. Unfortunately since 10.7, these files are encrypted by default and passwords (or other sensitive data) strings are no longer available. If Secure Virtual Memory is not enabled the user may use the `strings` command to find passwords and other non-encrypted strings that may be of use in an investigation.



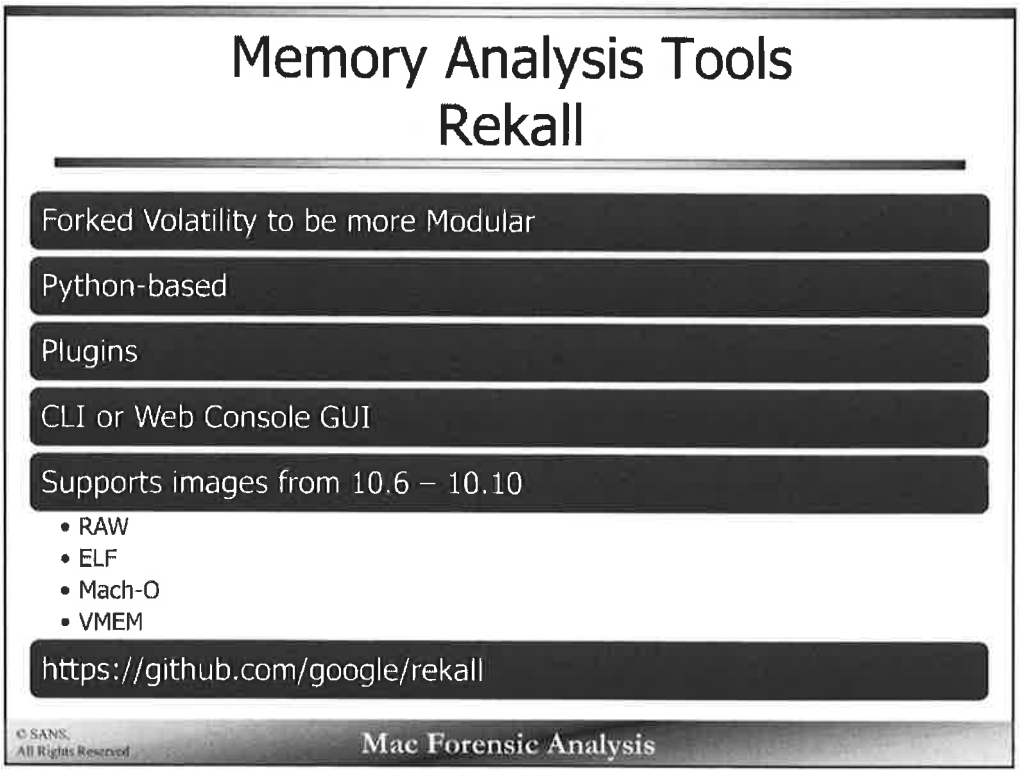
The de-facto standard in memory analysis, Volatility is a python-based tool can be used on any OS that supports Python.

While this tool also supports images from Windows, Linux, and Android systems', it started to support Mac systems in version 2.3. Images form 10.5 -10.9.x are supported with the image output formats listed above.

To make note of this again, download from source for the latest updates that have not yet been compiled in downloaded binaries or packages.

References:

https://github.com/volatilityfoundation/volatility



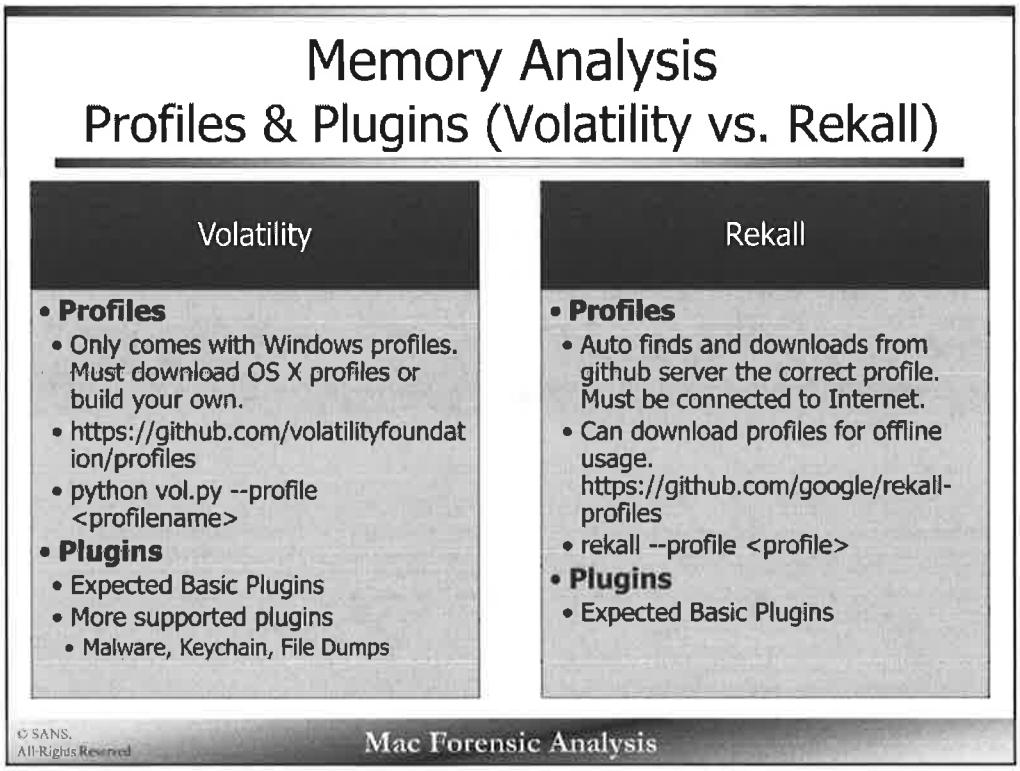
Rekall is a fork from Volatility so that it could be build with more modularity. As with Volatility it is python-based and has various plugins run to gather specific data types.

Rekall is based as a command-line utility, but can also be run from a web-based console GUI.

References:

http://www.rekall-forensic.com/

https://github.com/google/rekall



Profiles are what determine where certain data structures are located and how to look for them. For OS X it, a profile is required for each sub-version (i.e.: the profile for 10.94 is different than 10.9.3). Volatility and Rekall differ in how it get the profile for the memory image you are attempting to look at.

Volatility does not come with OS X profiles by default, you will need to download a profile from `https://github.com/volatilityfoundation/profiles` and place them in your /volatility/plugins/overlays/mac/ directory. To ensure you have them installed correctly run the command `python vol.py --info | grep Mac`. If no profile exists yet, you can build your own using the instructions provided here: `https://github.com/volatilityfoundation/volatility/wiki/Mac`

Rekall on the other hand will automatically determine and pull the correct profile from their repository here: `https://github.com/google/rekall-profiles`. You can clone their repository and use on an offline system if required using the instructions here: `http://www.rekall-forensic.com/faq.html`

The basic expected plugins exist for both Volatility and Rekall such as process listing, kext lists, open files, and network configuration. Volatility has done a better job with specialty plugins such as malware detection, Keychain key dumps, and file dumping.

References:

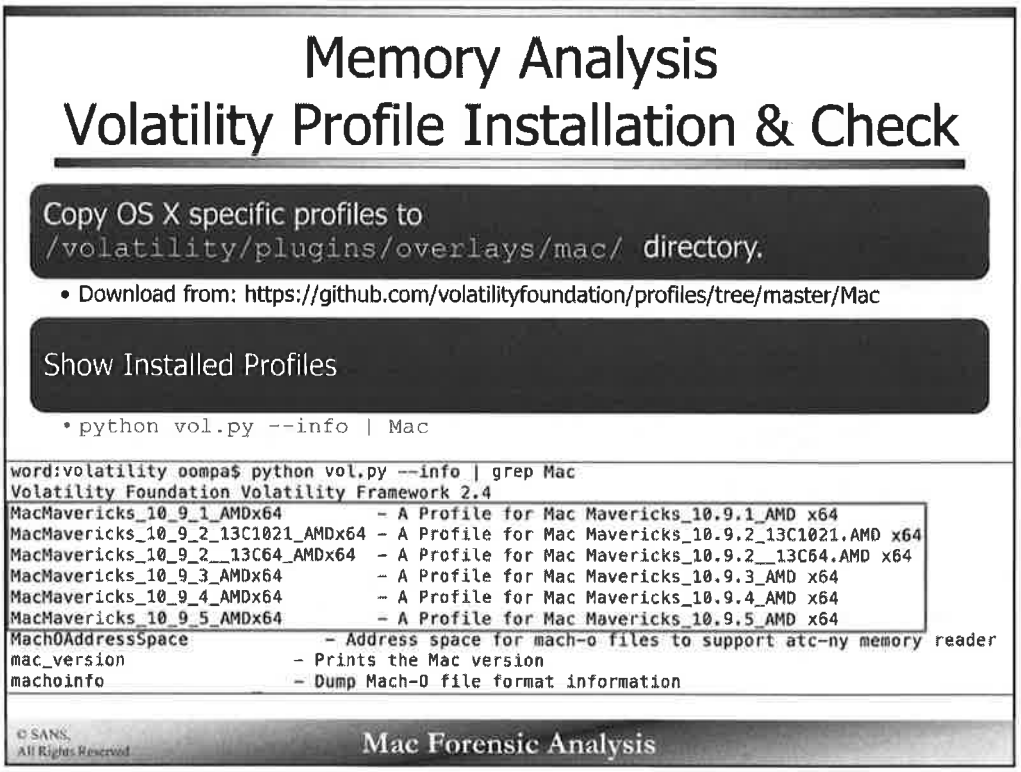
https://github.com/volatilityfoundation/volatility/wiki/Mac

https://github.com/volatilityfoundation/profiles

http://www.rekall-forensic.com/posts/2014-02-20-profile-selection.html

https://github.com/google/rekall-profiles/

http://www.rekall-forensic.com/faq.html



Volatility uses profiles to determine where certain data structures to present them to the user. These profiles are downloaded and installed in the `/volatility/plugins/overlays/mac/` directory in your volatility directory.

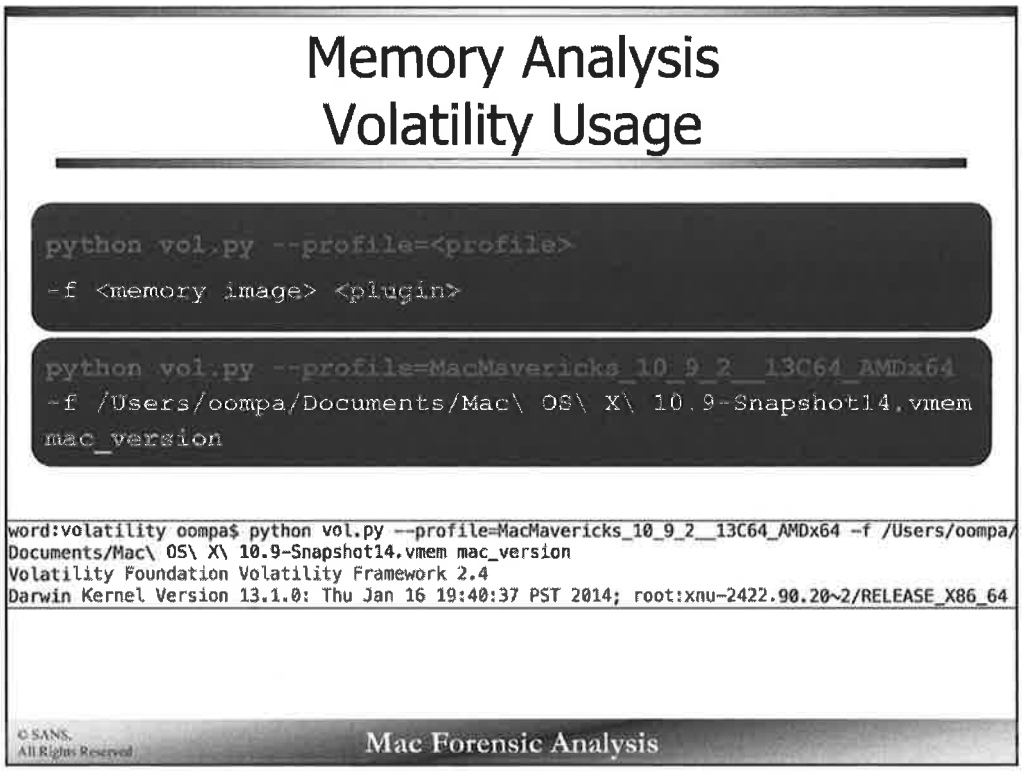
Once the files are copied into this directory, you can run the following command to determine if they are found by volatility.

```
python vol.py --info | Mac
```

As shown in the screenshot, you should be able to see the names of the profiles that you have just copied into the `/volatility/plugins/overlays/mac/` directory.

References:

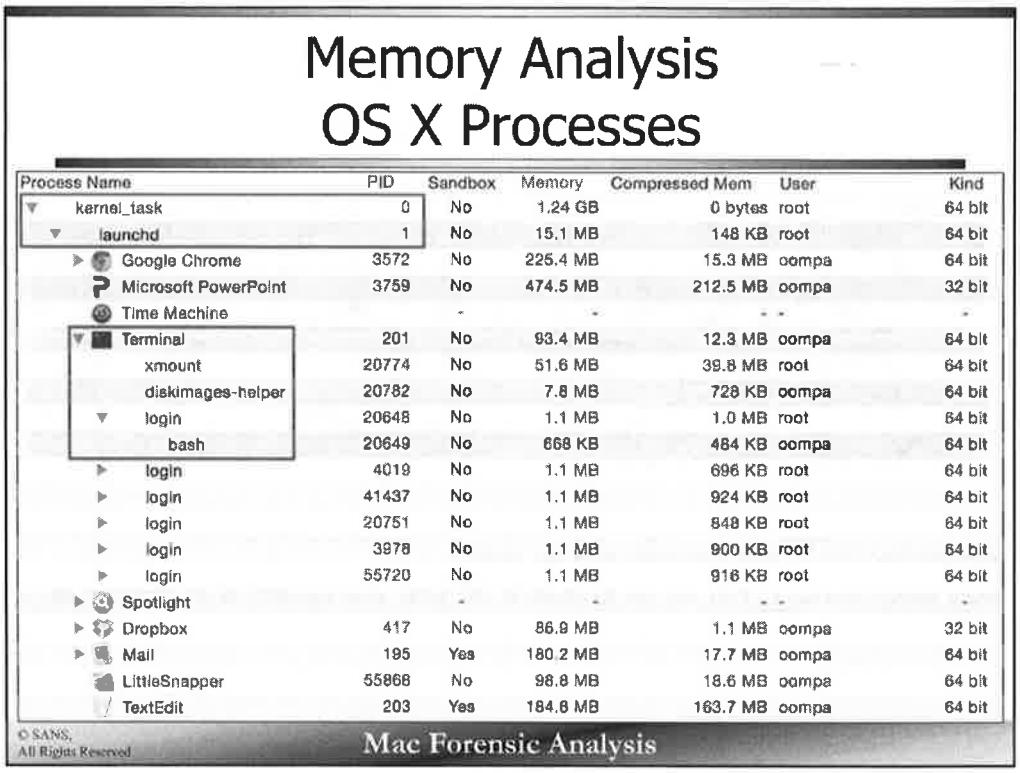
https://github.com/volatilityfoundation/volatility/wiki/Mac



Inputs to Volatility include at a minimum, a file path to a memory image and a plugin name to run. For Mac memory analysis, we'll need to specify a profile as described in the previous slide.

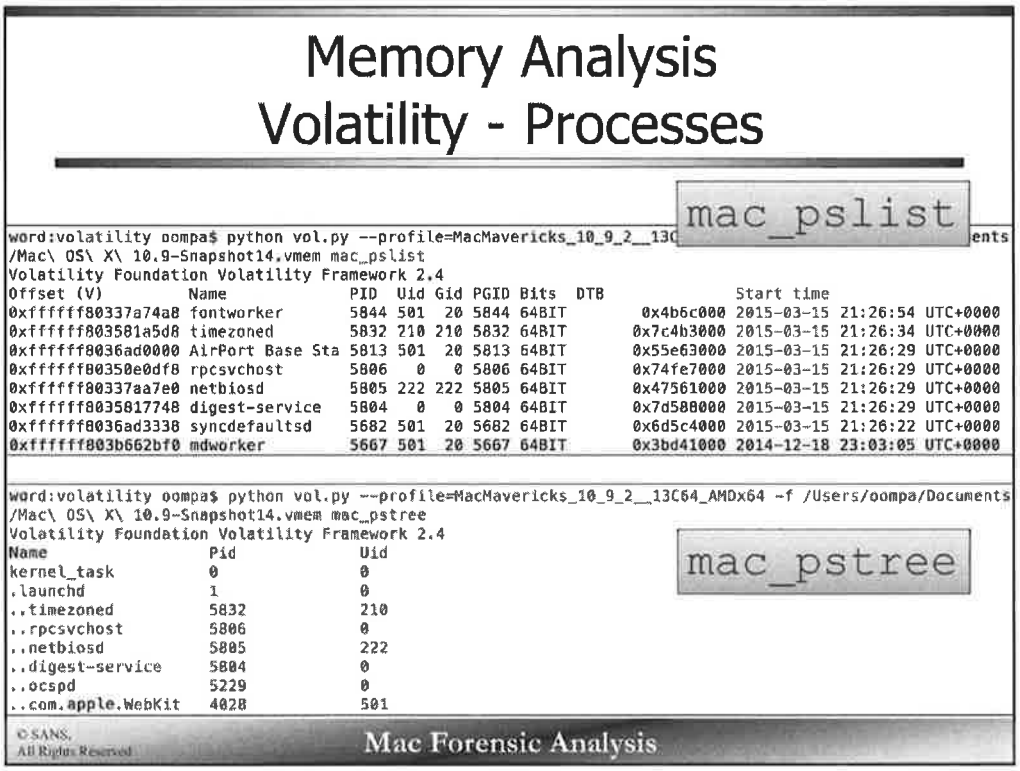
The example uses a profile for a specific build of 10.9.2 (Mavericks), the memory image is located in the user's Documents directory. This image is from a VMWare Fusion virtual machine snapshot (VMEM). The analyst is attempting to determine the basic system information for this memory image. This output is similar to the `uname -a` command.

```
python vol.py --profile=MacMavericks_10_9_2__13C64_AMDx64 -f
/Users/oompa/Documents/Mac\ OS\ X\ 10.9-Snapshot14.vmem mac_pslist
```



On a default system all processes will be a sub process of both `kernel_task` (always PID 0) and `launchd` (always PID 1). These parent processes are shown above in a screenshot taken from `Activity Monitor.app`.

Child processes for each application, agent, or daemon will be listed under `launchd`. In the screenshot example, the `Terminal` application as multiple `login` processes. Each of these `login` processes in turn has a child `bash` process.



Volatility has many different plugins. Two plugins for viewing system processes are `mac_pslist` and `mac_pstree`.

The top screenshot shows the `mac_pslist` plugin. This plugin prints the processes that were in use at the time the image was captured. Output information includes:

- Process Name
- Process ID (PID)
- UID/GID
- Process Architecture
- Process Start Time

The bottom screenshot shows the output of the `mac_pstree` plugin. This plugin shows the process name, PID, and UID in a tree-like format. This format allows child processes to be determine more quickly.

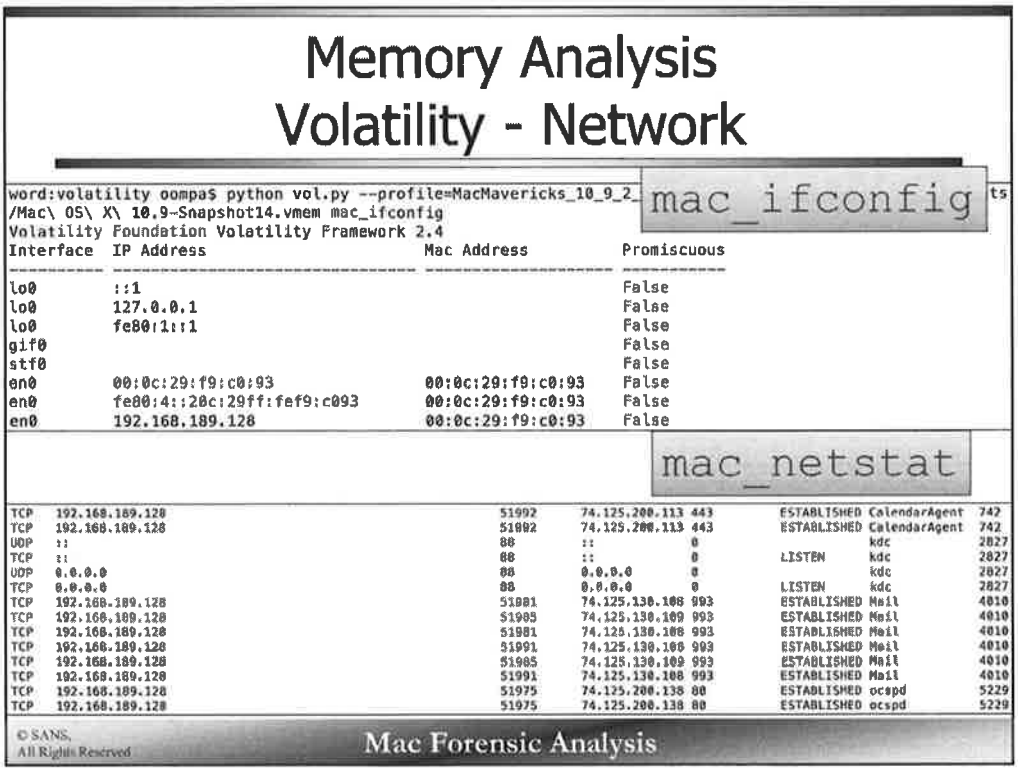
Reference:

https://github.com/volatilityfoundation/volatility/wiki/Mac-Command-Reference

```
word:volatility oompa$ python vol.py --profile=MacMavericks_10_9_2__13C64_AMDx64 -f /Users/oompa/Documents
/Mac\ OS\ X\ 10.9-Snapshot14.vmem mac_pslist
Volatility Foundation Volatility Framework 2.4
Offset (V)         Name                PID  Uid Gid PGID Bits  DTB               Start time
0xffffff80337a74a8 fontworker          5844 501  20 5844 64BIT        0x4b6c000 2015-03-15 21:26:54 UTC+0000
0xffffff803581a5d8 timezoned           5832 210 210 5832 64BIT       0x7c4b3000 2015-03-15 21:26:34 UTC+0000
0xffffff8036ad0000 AirPort Base Sta    5813 501  20 5813 64BIT       0x55e63000 2015-03-15 21:26:29 UTC+0000
0xffffff80350e0df8 rpcsvchost          5806   0   0 5806 64BIT       0x74fe7000 2015-03-15 21:26:29 UTC+0000
0xffffff80337aa7e0 netbiosd            5805 222 222 5805 64BIT       0x47561000 2015-03-15 21:26:29 UTC+0000
0xffffff8035817748 digest-service      5804   0   0 5804 64BIT       0x7d588000 2015-03-15 21:26:29 UTC+0000
0xffffff8036ad3338 syncdefaultsd       5682 501  20 5682 64BIT       0x6d5c4000 2015-03-15 21:26:22 UTC+0000
0xffffff803b662bf0 mdworker            5667 501  20 5667 64BIT       0x3bd41000 2014-12-18 23:03:05 UTC+0000
```



```
word:volatility oompa$ python vol.py --profile=MacMavericks_10_9_2__13C64_AMDx64 -f /Users/oompa/Documents
/Mac\ OS\ X\ 10.9-Snapshot14.vmem mac_pstree
Volatility Foundation Volatility Framework 2.4
Name                 Pid             Uid
kernel_task          0               0
.launchd             1               0
..timezoned          5832            210
..rpcsvchost         5806            0
..netbiosd           5805            222
..digest-service     5804            0
..ocspd              5229            0
..com.apple.WebKit   4028            501
```



Volatility can also show us volatile network information.

The `mac_ifconfig` provides us the current IP addresses and network interfaces of the system when the memory dump was captured.

The `mac_netstat` command output network connection IP, ports, status, and process affiliation.

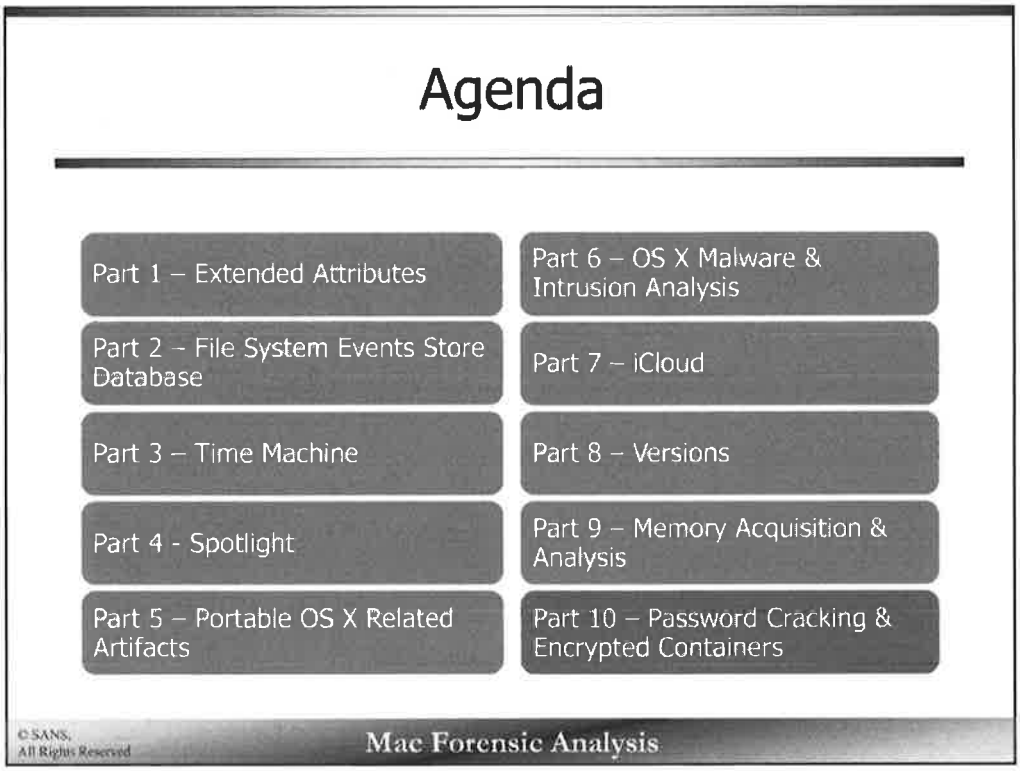
Reference:

https://github.com/volatilityfoundation/volatility/wiki/Mac-Command-Reference

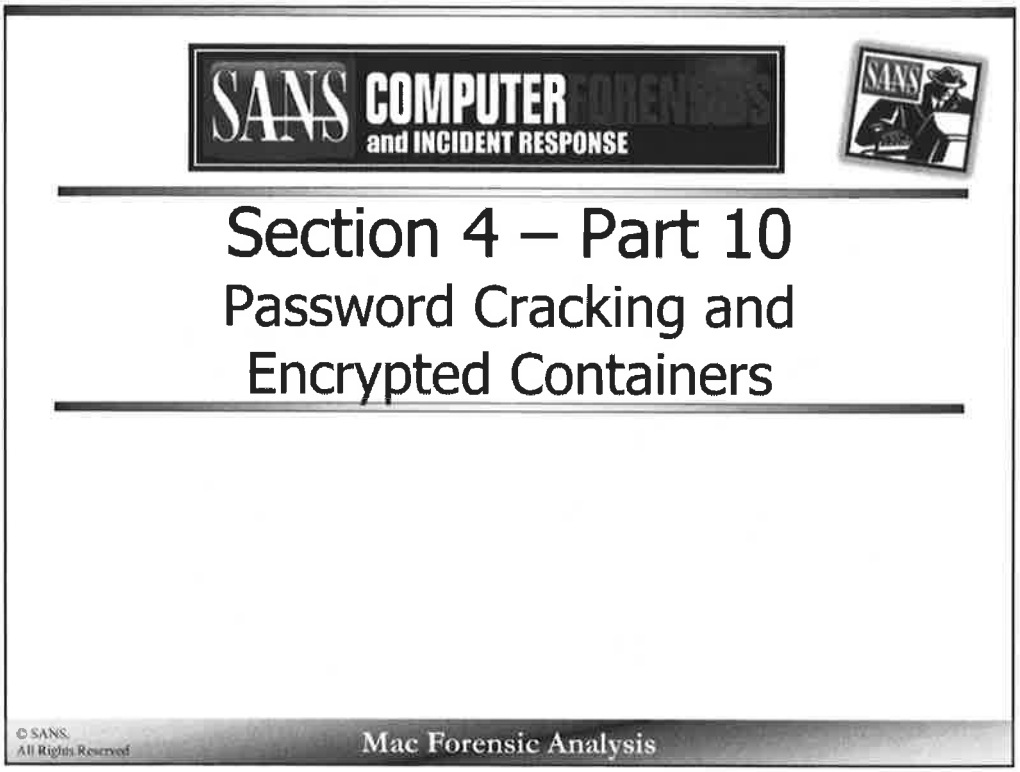
```
word:volatility oompa$ python vol.py --profile=MacMavericks_10_9_2__13C64_AMDx64 -f /Users/oompa/Documents
/Mac\ OS\ X\ 10.9-Snapshot14.vmem mac_ifconfig
Volatility Foundation Volatility Framework 2.4
Interface  IP Address                      Mac Address          Promiscuous
---------- ------------------------------- -------------------- -----------
lo0        ::1                                                  False
lo0        127.0.0.1                                            False
lo0        fe80:1::1                                            False
gif0                                                            False
stf0                                                            False
en0        00:0c:29:f9:c0:93               00:0c:29:f9:c0:93    False
en0        fe80:4::20c:29ff:fef9:c093      00:0c:29:f9:c0:93    False
en0        192.168.189.128                 00:0c:29:f9:c0:93    False
```



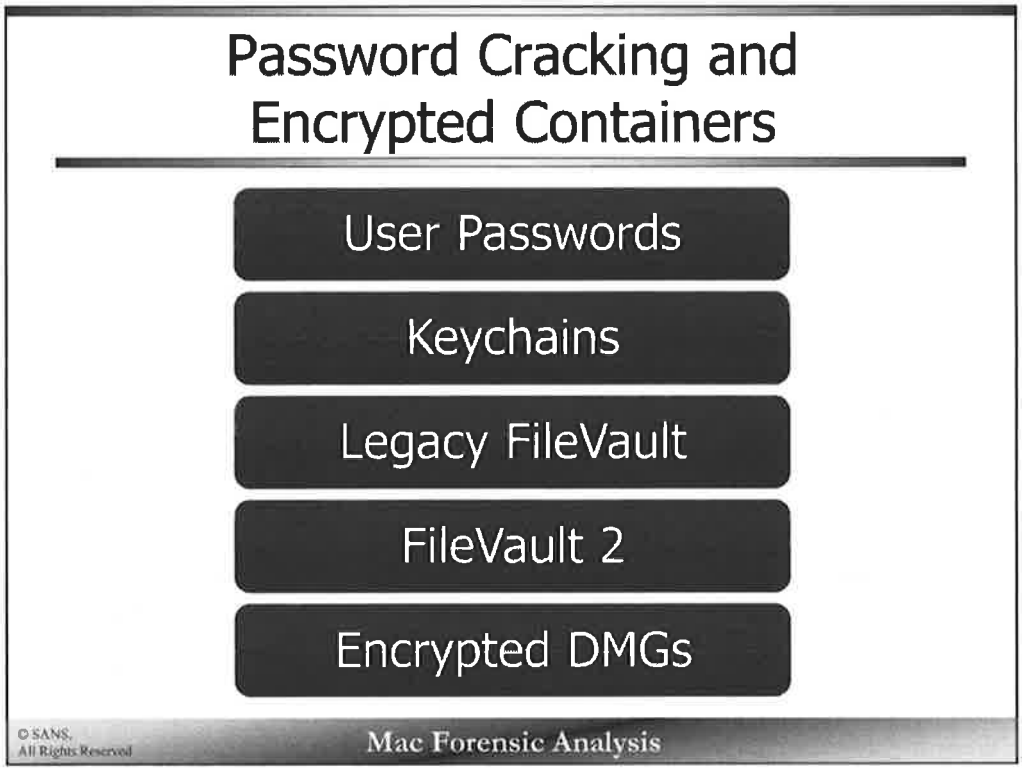
```
TCP   192.168.189.128                          51992      74.125.200.113 443          ESTABLISHED CalendarAgent  742
TCP   192.168.189.128                          51992      74.125.200.113 443          ESTABLISHED CalendarAgent  742
UDP   ::                                       88         ::             0                        kdc           2827
TCP   ::                                       88         ::             0            LISTEN      kdc           2827
UDP   0.0.0.0                                  88         0.0.0.0        0                        kdc           2827
TCP   0.0.0.0                                  88         0.0.0.0        0            LISTEN      kdc           2827
TCP   192.168.189.128                          51981      74.125.130.108 993          ESTABLISHED Mail          4010
TCP   192.168.189.128                          51985      74.125.130.109 993          ESTABLISHED Mail          4010
TCP   192.168.189.128                          51981      74.125.130.108 993          ESTABLISHED Mail          4010
TCP   192.168.189.128                          51991      74.125.130.108 993          ESTABLISHED Mail          4010
TCP   192.168.189.128                          51985      74.125.130.109 993          ESTABLISHED Mail          4010
TCP   192.168.189.128                          51991      74.125.130.108 993          ESTABLISHED Mail          4010
TCP   192.168.189.128                          51975      74.125.200.138 80           ESTABLISHED ocspd          5229
TCP   192.168.189.128                          51975      74.125.200.138 80           ESTABLISHED ocspd          5229
```



This page intentionally left blank.

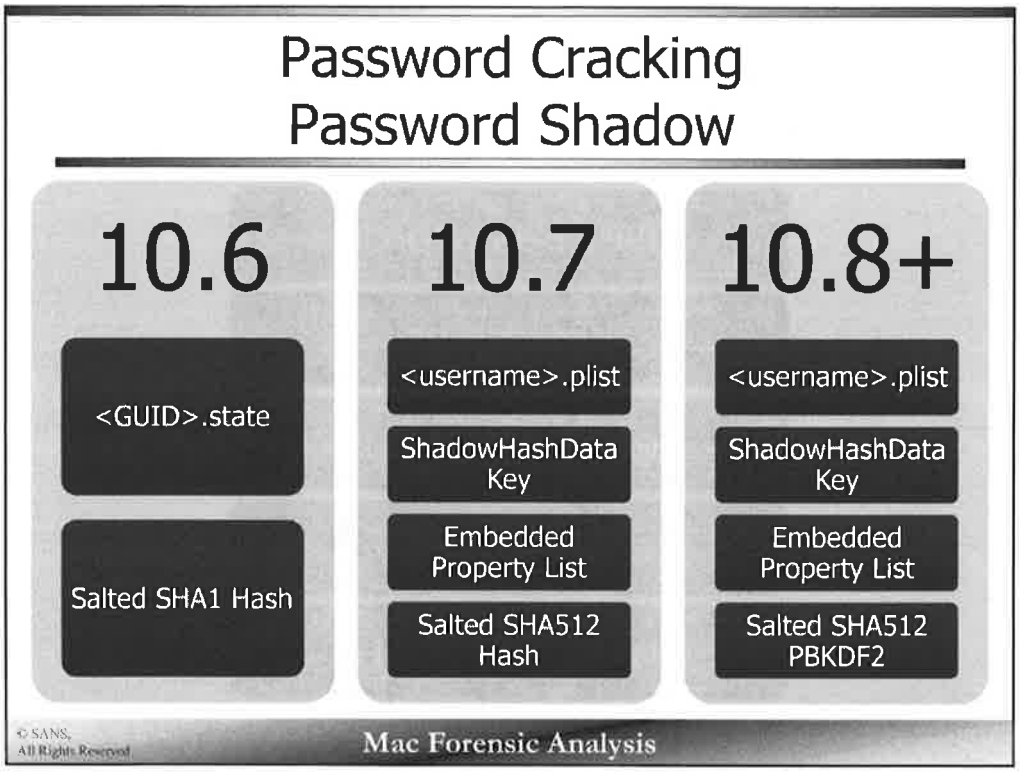


This page intentionally left blank.



An analyst might run into various encrypted containers during their analysis, such as FileVault volumes and encrypted disk image files.

These containers may be able to be cracked open by brute forcing the user passwords, or acquiring the passwords from the user's keychain files.



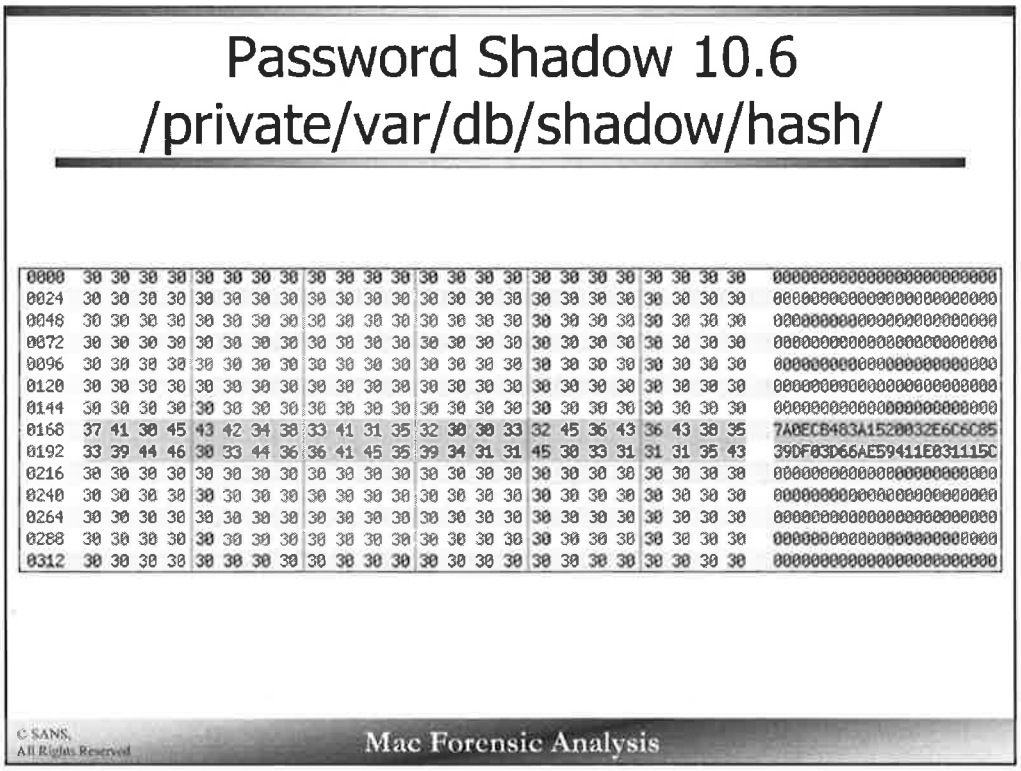
Like any Unix-based system, the user's password is stored in a password shadow file.

In 10.6 the location of the hash is stored in a the `<GUID>.state` file located in `/private/var/db/shadow/hash/` directory. This hash uses a salted SHA1 hash.

In 10.7+ systems the hash is now located in an embedded property list in the `ShadowHashDataKey` key in the user property list located in the `/private/var/db/dslocal/nodes/Default/users/` directory.

10.7 systems use a salted SHA512 hash, while 10.8+ systems use a salted SHA512 PBKDF2 hash.

The hashing algorithms have been increasingly more difficult to crack with every version of OS X. (We will see a good example of this in the related exercise!).

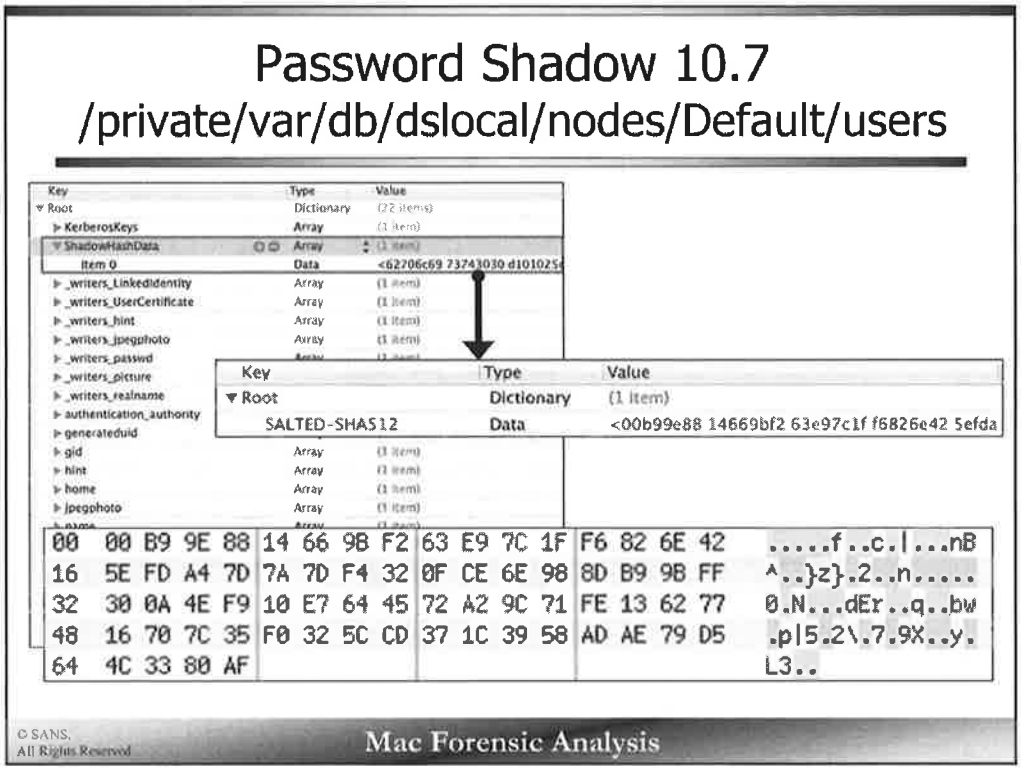


10.6 user passwords are stored in a file named with a specific GUID and the extension `.state`. These files are located in the `/private/var/db/shadow/hash/` directory.

The 48 byte SHA1 hash is found at offset 168 (highlighted above) in the file. Other hashes may be found at other offsets in this file.

The NT hash is at offset 0, and the LanMan hash can be found at offset 32, as detailed by the Defense in Depth blog at `http://www.defenceindepth.net/2009/12/cracking-os-x-passwords.html`. This will only occur if SMB/CIFS file sharing has been turned on.

0000	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0024	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0048	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0072	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0096	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0120	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0144	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0168	37	41	30	45	43	42	34	38	33	41	31	35	32	30	30	33	32	45	36	43	36	43	38	35	7A0ECB483A1520032E6C6C85
0192	33	39	44	46	30	33	44	36	36	41	45	35	39	34	31	31	45	30	33	31	31	31	35	43	39DF03D66AE59411E031115C
0216	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0240	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0264	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0288	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000
0312	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	00000000000000000000000000



Starting with 10.7 systems, the password hash is stored in a property list named for each user in the `/private/var/db/dslocal/nodes/Default/users/` directory.

The hash is stored in an embedded property list stored in the `ShadowHashDataKey`. This property list can be extracted and viewed as shown in the top screenshot.

The `SALTED-SHA512` key contains the combined salt and hash. The hex view shown in the bottom figure shows the hex output of the `SALTED-SHA512` key. The first four bytes (`0x00B99E88`) are the salt, while the rest of the bytes are the hash.

Reference:

Defense in Depth Blog:

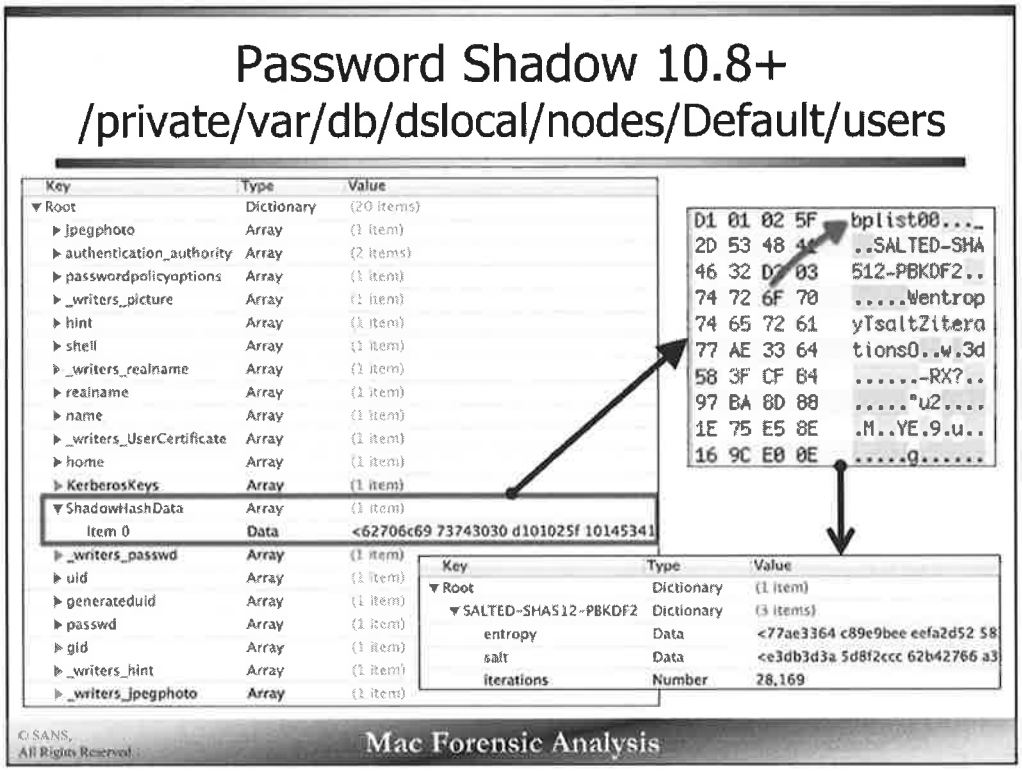
http://www.defenceindepth.net/2011/09/cracking-os-x-lion-passwords.html

Key	Type	Value
▼ Root	Dictionary	(22 items)
▶ KerberosKeys	Array	(1 item)
▼ ShadowHashData	Array	(1 item)
Item 0	Data	<62706c69 73743030 d101025c
▶ _writers_LinkedIdentity	Array	(1 item)
▶ _writers_UserCertificate	Array	(1 item)
▶ _writers_hint	Array	(1 item)
▶ _writers_jpegphoto	Array	(1 item)
▶ _writers_passwd	Array	(1 item)
▶ _writers_picture	Array	(1 item)
▶ passwd	Array	(1 item)
▶ passwordpolicyoptions	Array	(1 item)
▶ picture	Array	(1 item)
▶ realname	Array	(1 item)
▶ shell	Array	(1 item)
▶ uid	Array	(1 item)

```
00 00 B9 9E 88 14 66 9B F2 63 E9 7C 1F F6 82 6E 42   .....f..c.|...nB
16 5E FD A4 7D 7A 7D F4 32 0F CE 6E 98 8D B9 9B FF   ^..}z}.2..n.....
32 30 0A 4E F9 10 E7 64 45 72 A2 9C 71 FE 13 62 77   0.N...dEr..q..bw
48 16 70 7C 35 F0 32 5C CD 37 1C 39 58 AD AE 79 D5   .p|5.2\.7.9X..y.
64 4C 33 80 AF                                       L3..
```

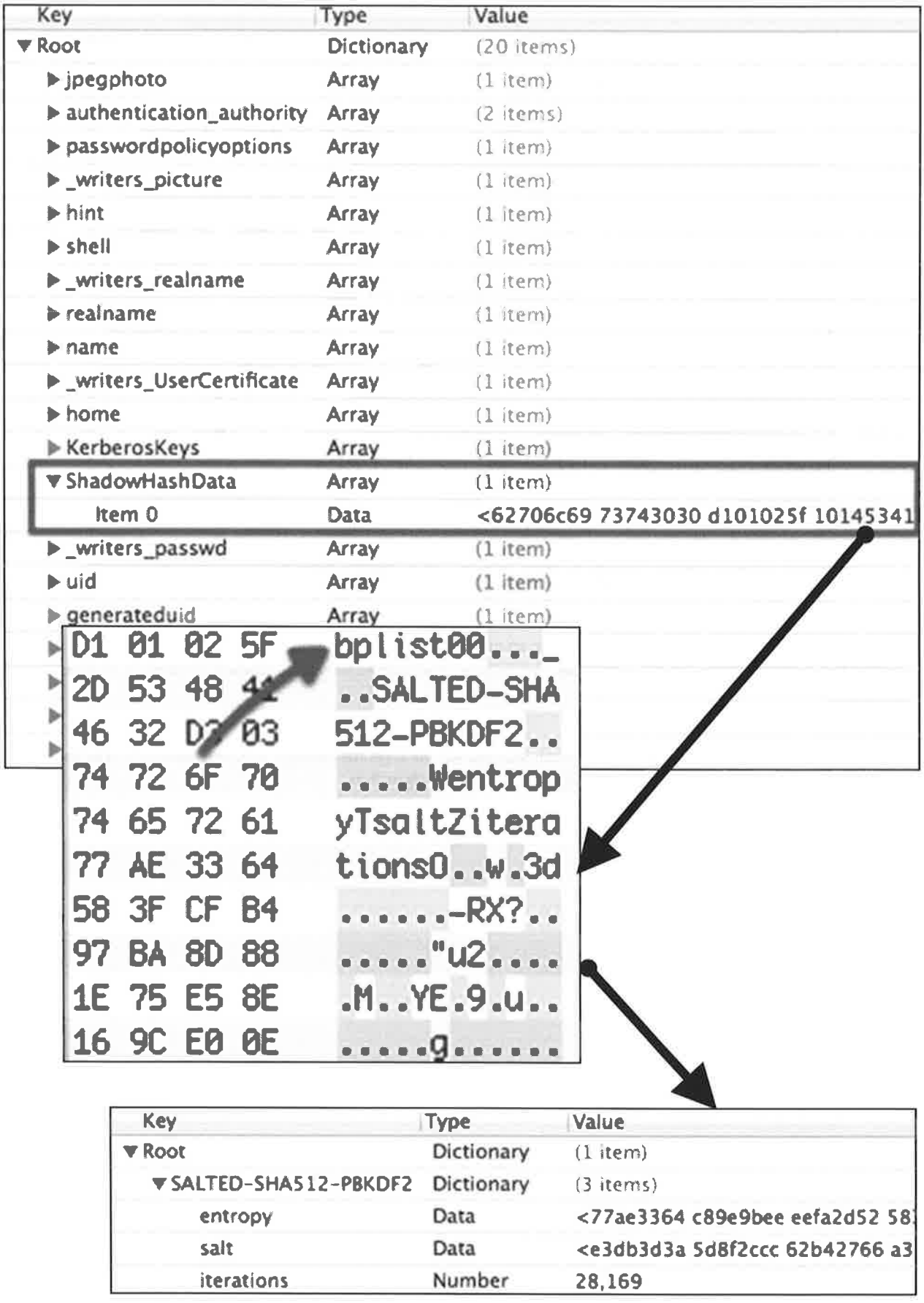
Key	Type	Value
▼ Root	Dictionary	(1 item)
SALTED-SHA512	Data	<00b99e88 14669bf2 63e97c1f f6826e42 5efda

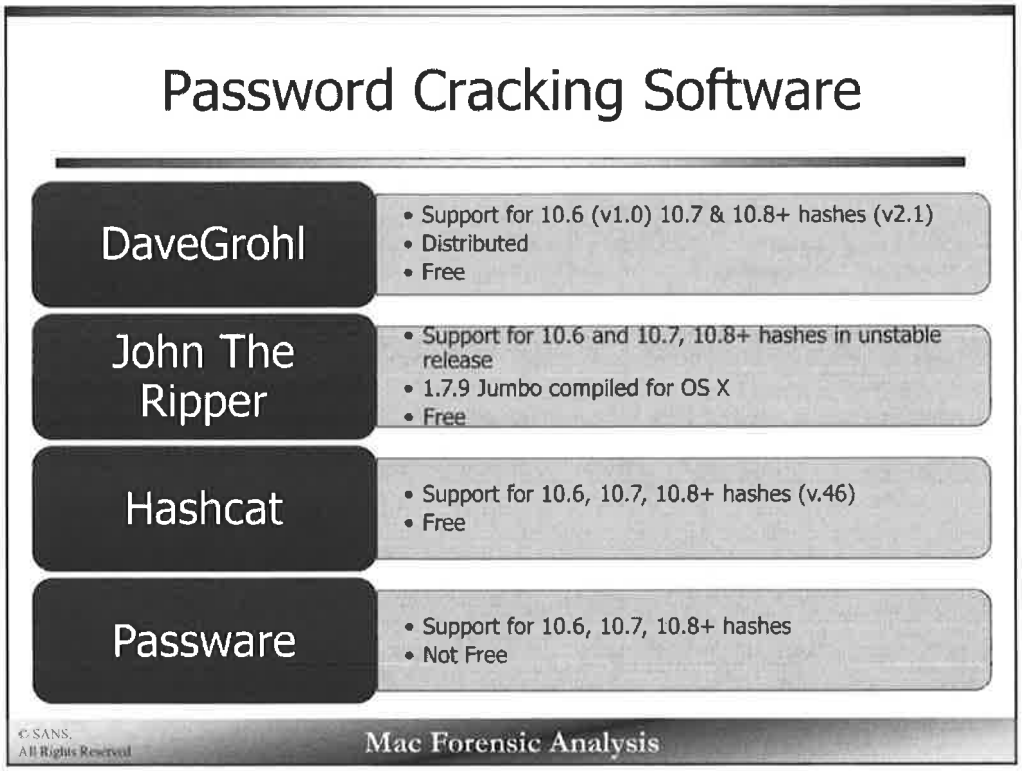
224



Similar to 10.7, 10.8+ systems store the hash in an embedded property list within the `ShadowHashData` key within the user's property list file located in `/private/var/db/dslocal/nodes/Default/users/` directory.

This property list can be extracted and viewed. We can see the keys for entropy, salt, and iterations. The hash algorithm differs from 10.7 as it uses the SHA512 PBKDF2 algorithm.


Key
Type
Value
Root
Dictionary
(20 items)
jpegphoto
Array
(1 item)
authentication_authority
Array
(2 items)
passwordpolicyoptions
Array
(1 item)
_writers_picture
Array
(1 item)
hint
Array
(1 item)
shell
Array
(1 item)
_writers_realname
Array
(1 item)
realname
Array
(1 item)
name
Array
(1 item)
_writers_UserCertificate
Array
(1 item)
home
Array
(1 item)
KerberosKeys
Array
(1 item)
ShadowHashData
Array
(1 item)
Item 0
Data
<62706c69 73743030 d101025f 10145341
_writers_passwd
Array
(1 item)
uid
Array
(1 item)
generateduid
Array
(1 item)
D1 01 02 5F
bplist00..._
2D 53 48 41
..SALTED-SHA
46 32 D2 03
512-PBKDF2..
74 72 6F 70
.....Wentrop
74 65 72 61
yTsaltZitera
77 AE 33 64
tionsO..w.3d
58 3F CF B4
.....-RX?..
97 BA 8D 88
....."u2....
1E 75 E5 8E
.M..YE.9.u..
16 9C E0 0E
.....g.....
Key
Type
Value
Root
Dictionary
(1 item)
SALTED-SHA512-PBKDF2
Dictionary
(3 items)
entropy
Data
<77ae3364 c89e9bee eefa2d52 58
salt
Data
<e3db3d3a 5d8f2ccc 62b42766 a3
iterations
Number
28,169



Various password cracking software is available. These programs range from free to quite expensive. The software programs listed above work for each of the different password hashing algorithms used in 10.6 – 10.8+.

References:

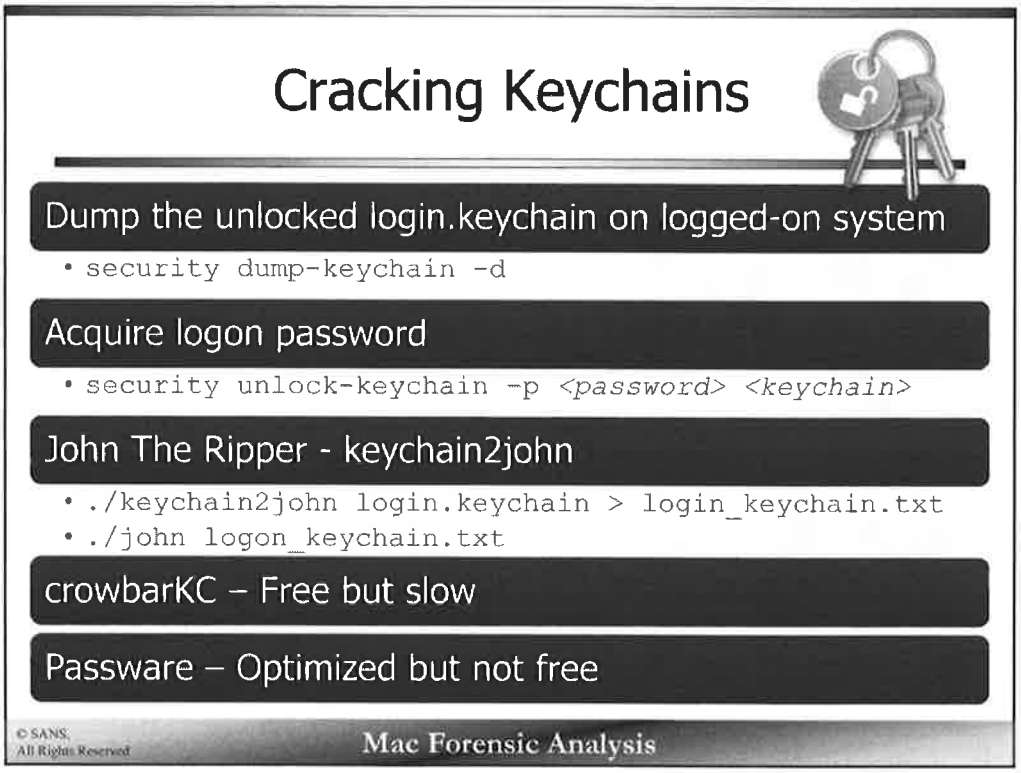
Dave Grohl - http://www.davegrohl.org/

John The Ripper

- http://openwall.info/wiki/john/custom-builds#Compiled-for-Mac-OS-X
- http://download.openwall.net/pub/projects/john/contrib/macosx/

Hashcat - http://hashcat.net/hashcat/

Passwarc - http://www.lostpassword.com/kit-forensic.htm



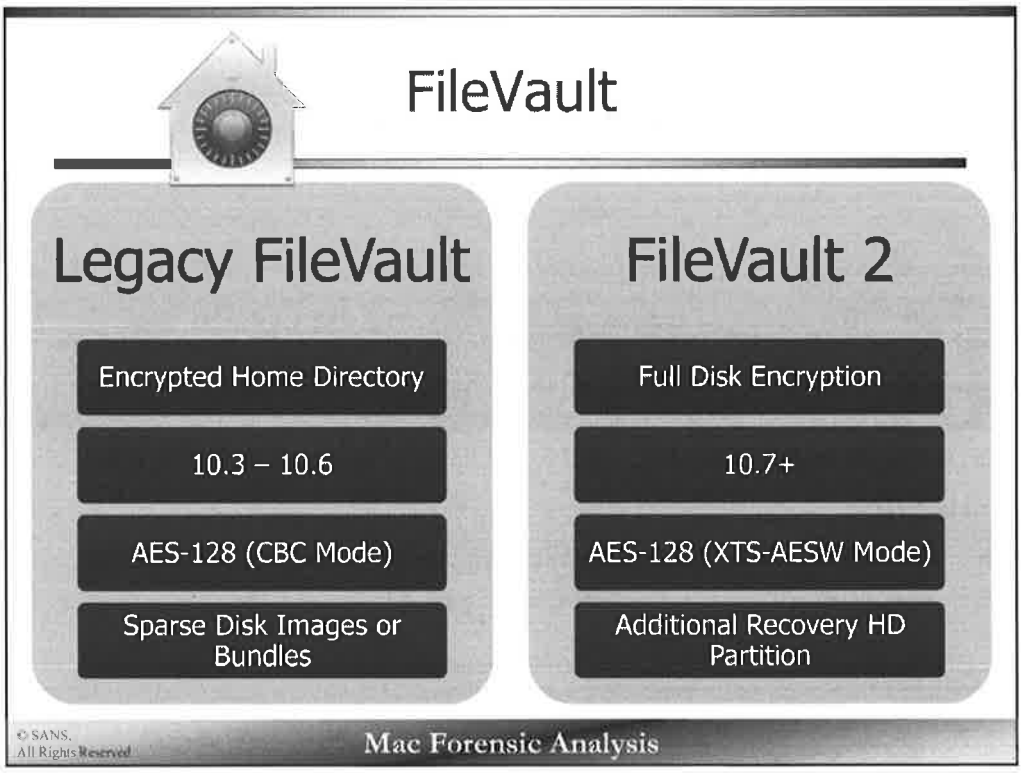
Keychains can be useful to gather a variety of passwords. Many people reuse passwords or parts of passwords, this can be used to shorten the wordlist to brute force other passwords.

On an already logged-on system, an investigator is able to dump the passwords (usually without a password) of the currently logged on user's `login.keychain`. The `security` command used with the `dump-keychain -d` option can be used to do this. This command produces quite a bit of output, so redirection to a file is recommended for later viewing.

If a user's logon password can be acquired, an analyst can usually use that to access the keychain by using the `unlock-keychain` option to the `security` command. "Usually" being a key term - the user's password is often used as the default password to a user's keychain, however a user can change this at any time.

The well known password cracking program John the Ripper can be used to brute force keychain passwords. Specially compiled versions (at least 1.7.9 Jumbo 6) of JTR will have the program `keychain2john` that is used to dump the password hash from the keychain file and into JTR format. This output can be saved to a file (`login_keychain.txt` for example), and used with the john program. It should be noted, this may take a LONG time to crack and highly depends on the configuration, JTR wordlists, and complexity of the password. A good resource for compiling and configuring JTR for this purpose can be found here: http://easymactips.blogspot.com/2012/09/john-ripper-tutorial-examples-and.html

Other programs such as crowbarKC (available at http://www.georgestarcher.com/?page_id=256) and Passware (available at http://www.lostpassword.com/) can be used to brute force keychains. CrowbarKC is a free program but is quite slow (even compared with JTR). Passware is more optimized, but it is not free.



FileVault (now call Legacy FileVault) was introduced in version 10.3. FileVault, if implemented, encrypts the home directory of a user. All other domains and user directories remain unencrypted. The encrypted home directory files are stored in a sparse disk image, or a sparse bundle.

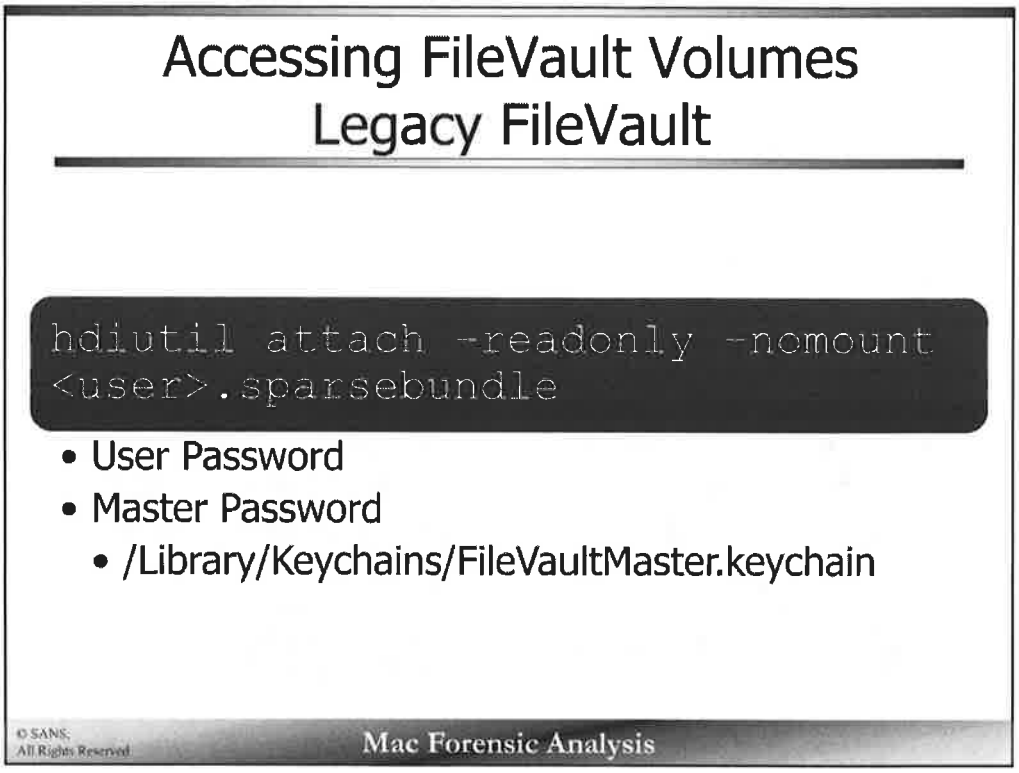
FileVault 2 was introduced in Mac OS X Lion, and encrypts the whole disk, except for the EFI and Recovery partitions.

Different versions of AES encryption are used in each FileVault implementation. Like password hash algorithms, Apple continues to improve the use of its encryption to create a stronger container.

Reference:

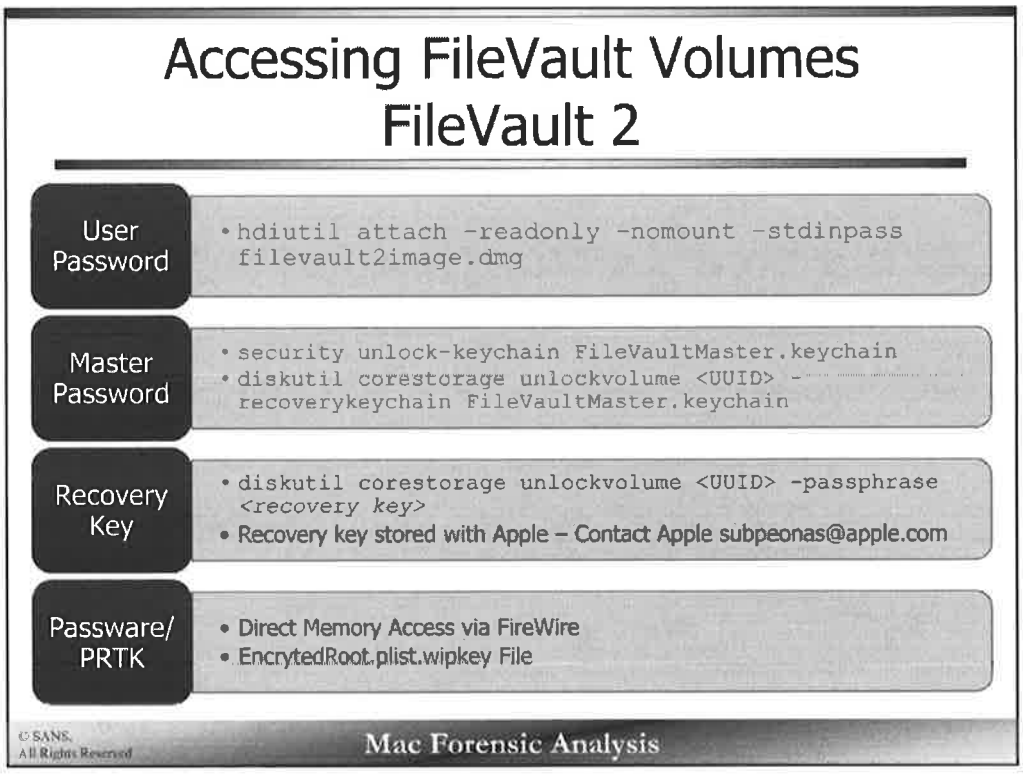
OS X: About FileVault 2

[http://support.apple.com/kb/HT4790]



Legacy FileVault `sparsebundles` can be cracked using the user's logon password or by using the FileVault Master password. The master password is stored in the `FileVaultMaster.keychain` file located in the `/Library/Keychains/` directory.

To mount the `sparsebundle` on a Mac to be able to image the unencrypted version we can use the `hdiutil` command. This command used with the attach verb, along with the `-readonly -nomount` options will prompt a user for a password, and will mount the volume on a `/dev/` device. This can then be imaged just as any other disk would be.



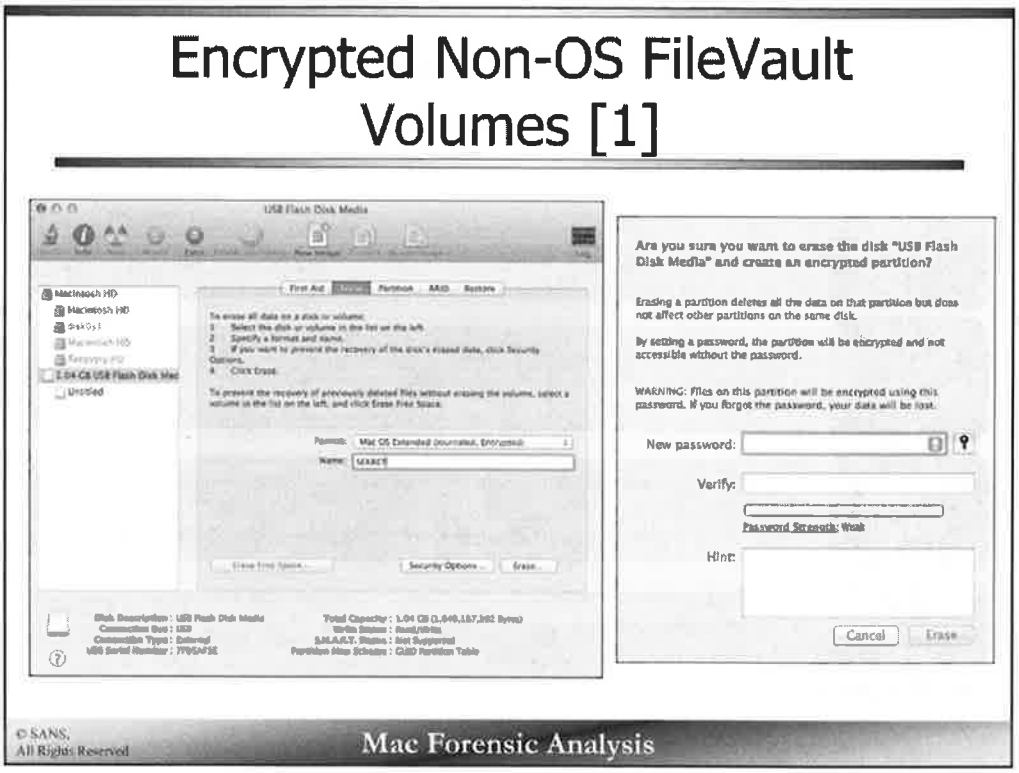
FileVault 2 volumes can be accessed by using the user's password, the master password, a recovery key, or by direct memory access using a program like Passware.

References:

http://derflounder.wordpress.com/2011/11/23/using-the-command-line-to-unlock-or-decrypt-your-filevault-2-encrypted-boot-drive/

http://www.lostpassword.com/hdd-decryption.htm

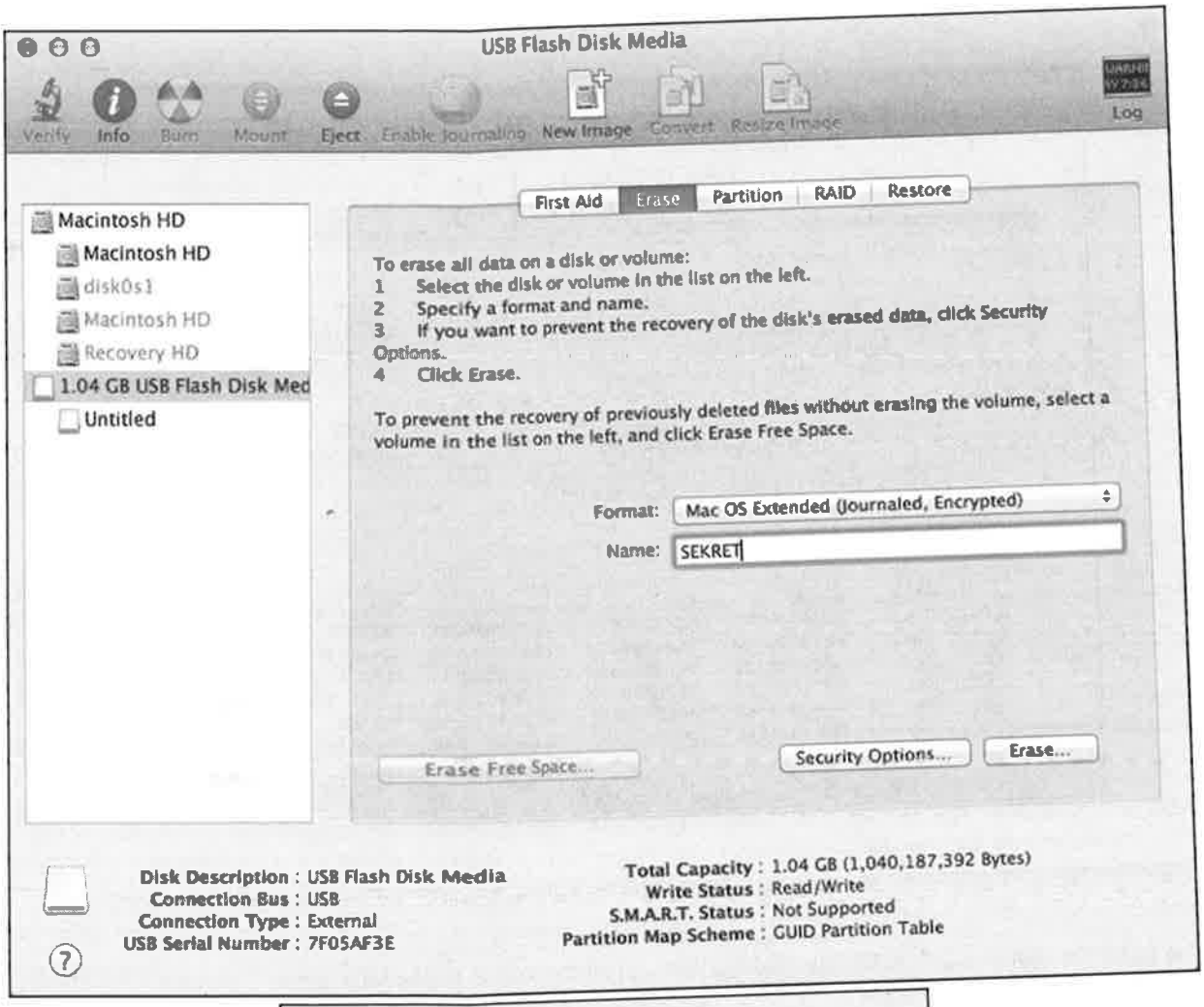
https://support.apple.com/kb/HT4790

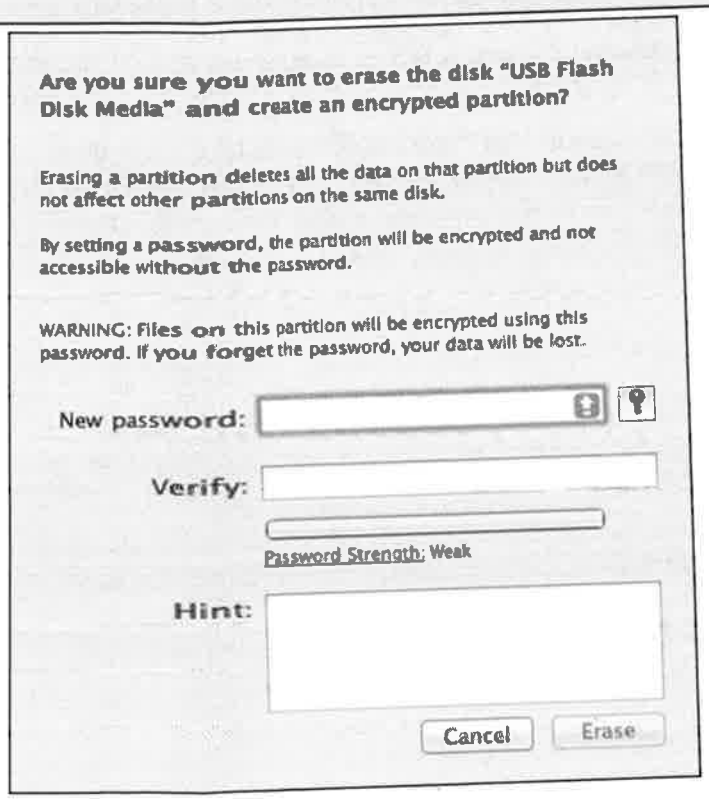


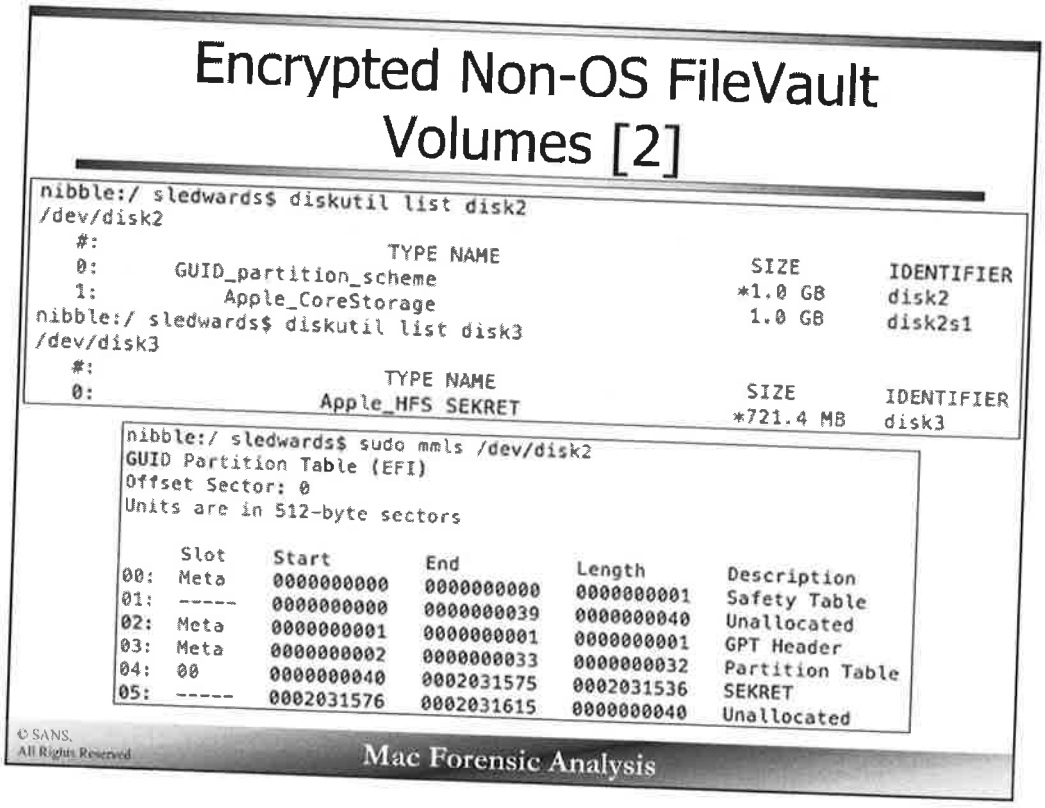
Other volumes may also be encrypted using FileVault. These volumes may be external hard drives, thumb drives, or other types of disk storage.

The left screenshot shows the Disk Utility application with a 1GB thumb drive attached. To encrypt this thumb drive, a user will need to go to the "`Erase`" tab and choose the "`Mac OS Extended (Journaled, Encrypted)`" in the Format options.

The right screenshot shows the pop-up window the user will receive when the "`Erase...`" procedure is started. This window allows the user to input a password (and password hint) for the volume.


USB Flash Disk Media
Verify
Info
Burn
Mount
Eject
Enable Journaling
New Image
Convert
Resize Image
Log
Macintosh HD
Macintosh HD
disk0s1
Macintosh HD
Recovery HD
1.04 GB USB Flash Disk Med
Untitled
First Aid
Erase
Partition
RAID
Restore
To erase all data on a disk or volume:
1 Select the disk or volume in the list on the left.
2 Specify a format and name.
3 If you want to prevent the recovery of the disk's erased data, click Security Options.
4 Click Erase.
To prevent the recovery of previously deleted files without erasing the volume, select a volume in the list on the left, and click Erase Free Space.
Format: Mac OS Extended (Journaled, Encrypted)
Name: SEKRET
Erase Free Space...
Security Options...
Erase...
Disk Description : USB Flash Disk Media
Connection Bus : USB
Connection Type : External
USB Serial Number : 7F05AF3E
Total Capacity : 1.04 GB (1,040,187,392 Bytes)
Write Status : Read/Write
S.M.A.R.T. Status : Not Supported
Partition Map Scheme : GUID Partition Table


Are you sure you want to erase the disk "USB Flash Disk Media" and create an encrypted partition?
Erasing a partition deletes all the data on that partition but does not affect other partitions on the same disk.
By setting a password, the partition will be encrypted and not accessible without the password.
WARNING: Files on this partition will be encrypted using this password. If you forget the password, your data will be lost.
New password:
Verify:
Password Strength: Weak
Hint:
Cancel
Erase



These encrypted volumes use the `CoreStorage` implementation we saw earlier in the course.

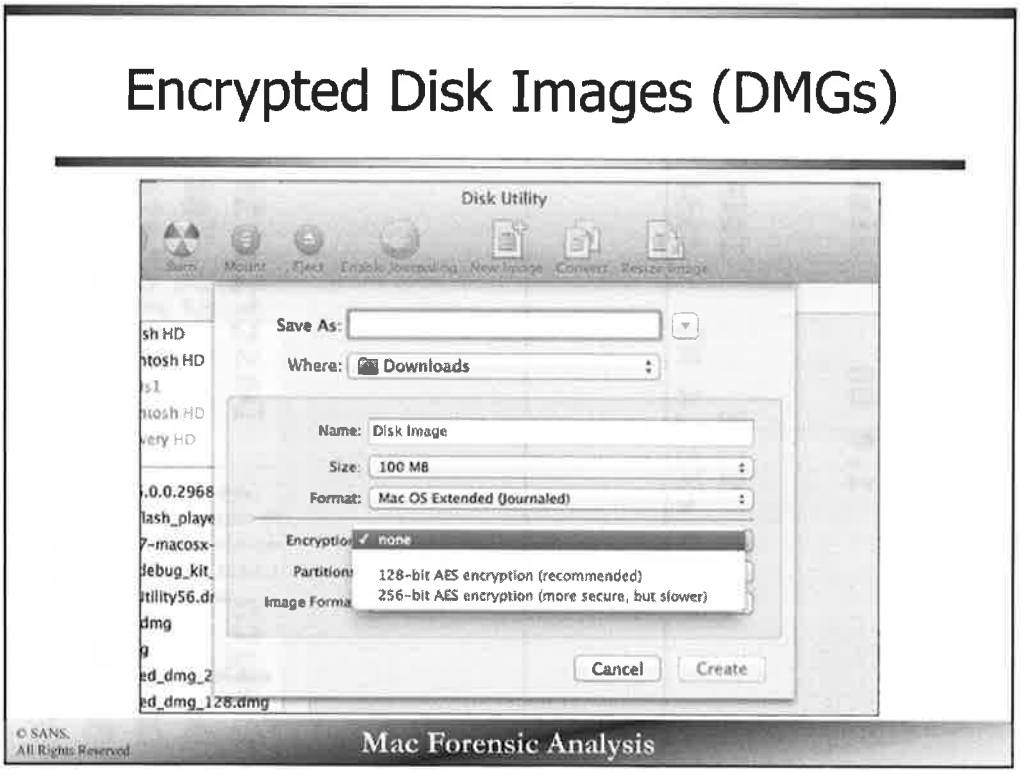
The top screenshot shows the 1GB thumb drive that we just encrypted using Disk Utility. `/dev/disk2` shows the physical thumb drive with the `CoreStorage` partition, while `/dev/disk3` shows the mounted encrypted volume called "`SEKRET`". This volume would only be available if the password were given correctly.

The bottom screenshot is an example of what `/dev/disk2` looks like using the `mmls` command from Sleuthkit. We can see the main partition is likely "`SEKRET`", but we are unable to tell if it is an encrypted volume.

```
nibble:/ sledwards$ diskutil list disk2
/dev/disk2
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *1.0 GB     disk2
   1:          Apple_CoreStorage                         1.0 GB     disk2s1
nibble:/ sledwards$ diskutil list disk3
/dev/disk3
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:                  Apple_HFS SEKRET                  *721.4 MB   disk3
```

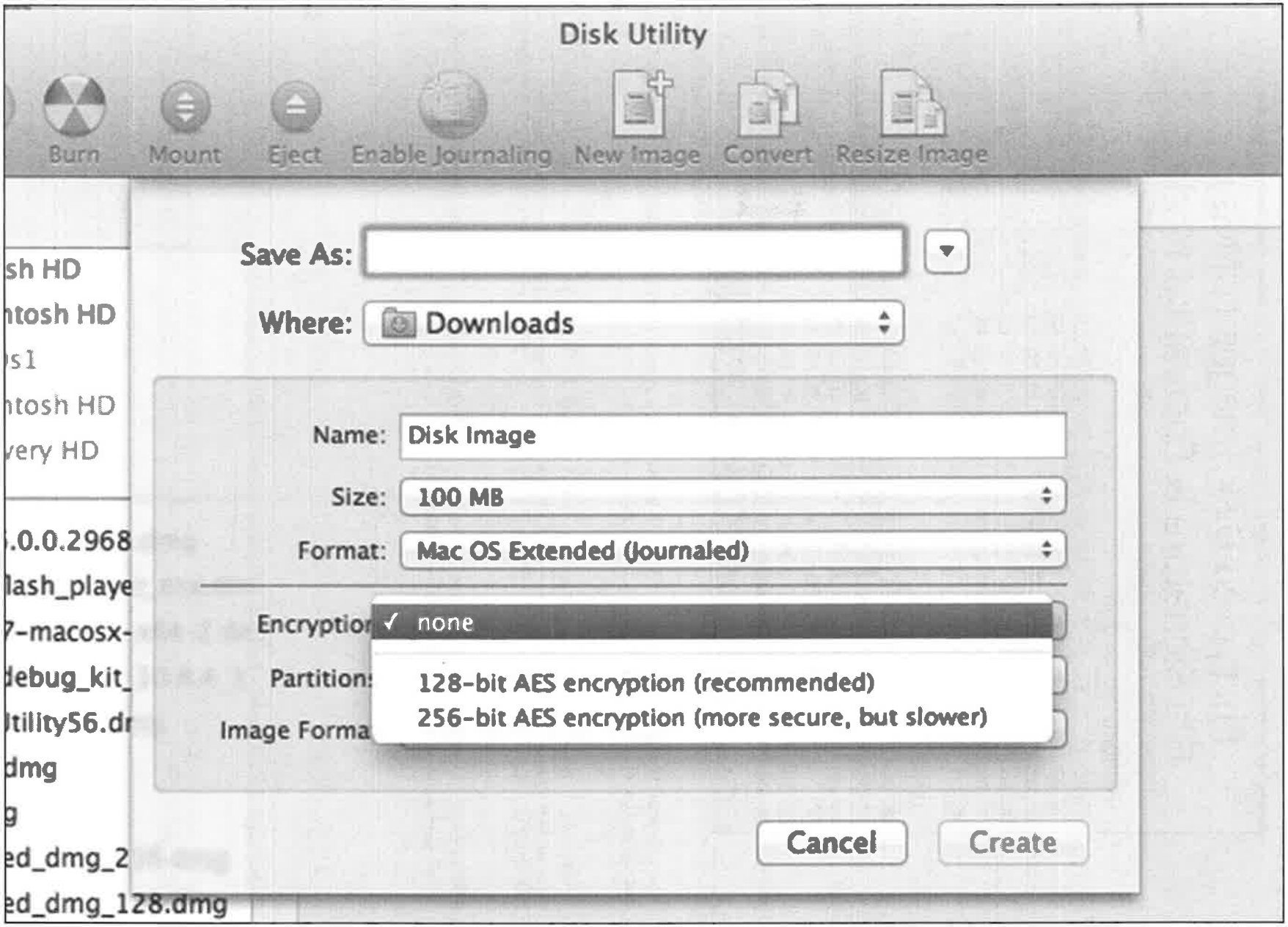


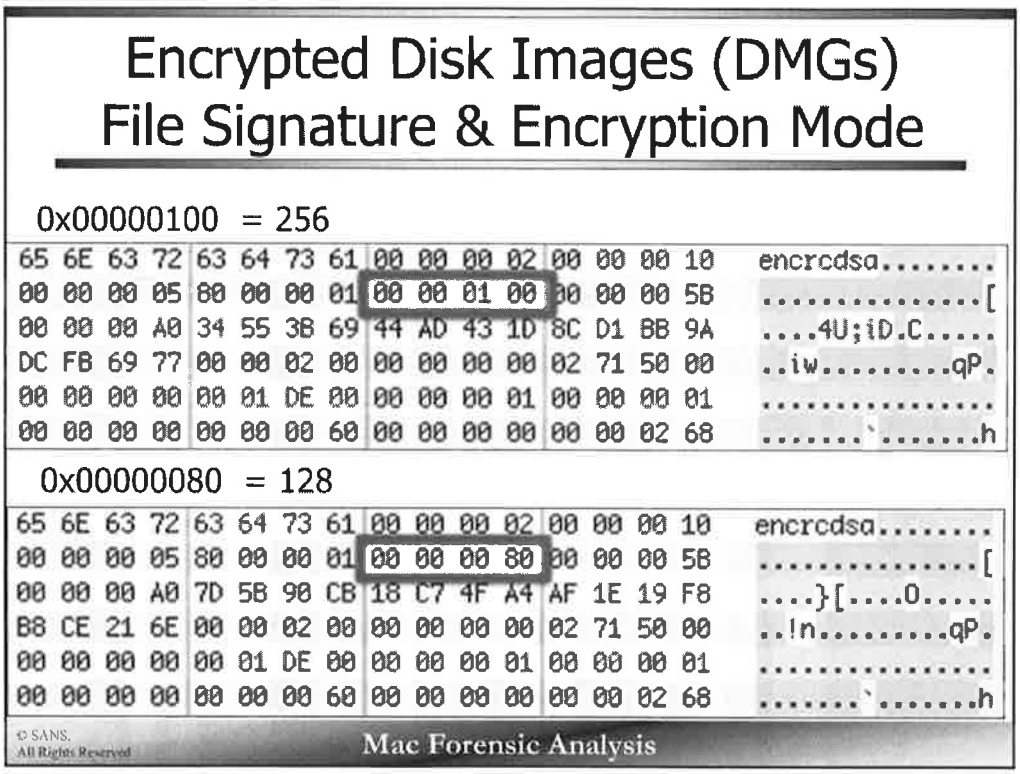
```
nibble:/ sledwards$ sudo mmls /dev/disk2
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  Meta    0000000000   0000000000   0000000001   Safety Table
01:  -----   0000000000   0000000039   0000000040   Unallocated
02:  Meta    0000000001   0000000001   0000000001   GPT Header
03:  Meta    0000000002   0000000033   0000000032   Partition Table
04:  00      0000000040   0002031575   0002031536   SEKRET
05:  -----   0002031576   0002031615   0000000040   Unallocated
```



Disk Images (or DMG files) can be created using the native OS X tool Disk Utility or via the command line using `hdiutil`. These disk images are encapsulated files that can be created in any size and can contain any data you want. Other options include various file system formats (HFS+, FAT, ExFAT), partition choices (CD/DVD, GUID, MBR), and image formats (sparse bundle, sparse disk, read/write, CD/DVD master).

While disk images do not have to be encrypted, the user does have the ability to encrypt them using AES encryption with 128 or 256-bit keys. When these disk images are created a popup window allows a user to input a password.


Disk Utility
Burn
Mount
Eject
Enable Journaling
New Image
Convert
Resize Image
Save As:
Where:
Downloads
Name:
Disk Image
Size:
100 MB
Format:
Mac OS Extended (Journaled)
Encryption:
✓ none
128-bit AES encryption (recommended)
256-bit AES encryption (more secure, but slower)
Cancel
Create



Two encrypted DMGs were created, one with 256-bit AES in encryption (top) and another with 128-bit AES encryption (bottom). Each DMG file has the file signature "`encrcdsa`".

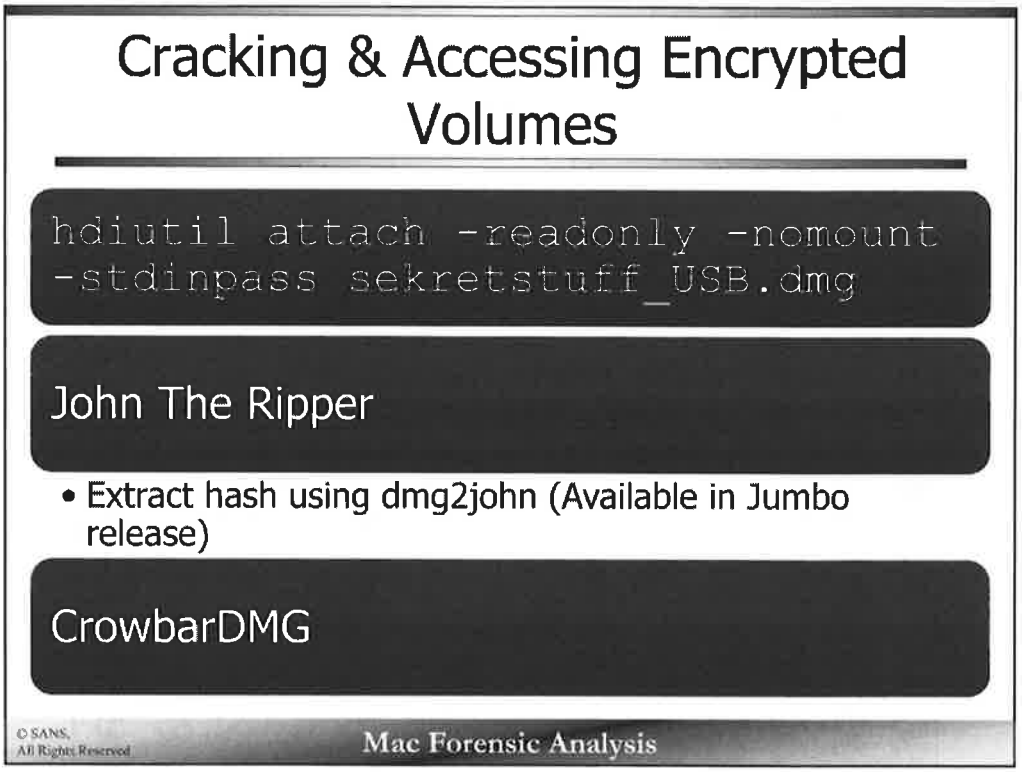
An easy way to tell what encryption strength was used to encrypt the DMG is to look at the 4 bytes at offset `24`, each are highlighted in the red boxes above. The top calculates out (big-endian) to 256, while the bottom calculates to 128.

Older `sparsedisk` image files (Version 1) may have the string "`cdsaencr`" in the signature at the very end of the volume, rather than the beginning as shown above.

```
65 6E 63 72 63 64 73 61 00 00 00 02 00 00 00 10   encrcdsa........
00 00 00 05 80 00 00 01 00 00 01 00 00 00 00 5B   ...............[
00 00 00 A0 34 55 3B 69 44 AD 43 1D 8C D1 BB 9A   ....4U;iD.C.....
DC FB 69 77 00 00 02 00 00 00 00 00 02 71 50 00   ..iw.........qP.
00 00 00 00 00 01 DE 00 00 00 00 01 00 00 00 01   ................
00 00 00 00 00 00 00 60 00 00 00 00 00 00 02 68   .......`.......h
```



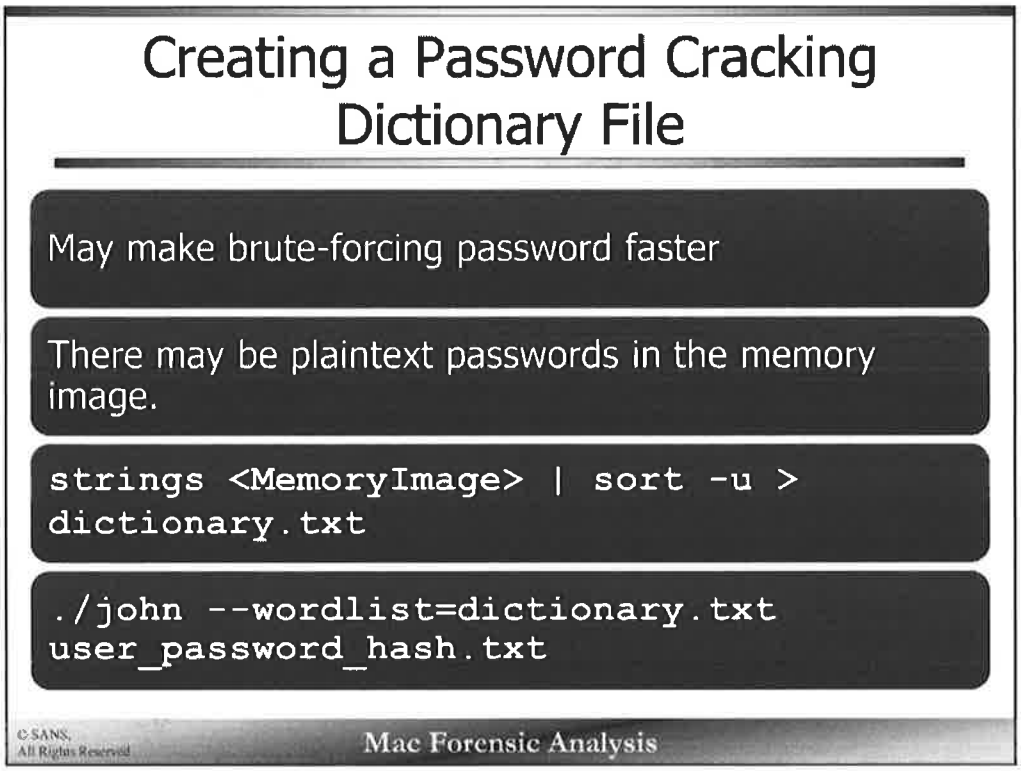
```
65 6E 63 72 63 64 73 61 00 00 00 02 00 00 00 10   encrcdsa........
00 00 00 05 80 00 00 01 00 00 00 80 00 00 00 5B   ...............[
00 00 00 A0 7D 5B 90 CB 18 C7 4F A4 AF 1E 19 F8   ....}[....O.....
B8 CE 21 6E 00 00 02 00 00 00 00 00 02 71 50 00   ..!n.........qP.
00 00 00 00 00 01 DE 00 00 00 00 01 00 00 00 01   ................
00 00 00 00 00 00 00 60 00 00 00 00 00 00 02 68   .......`.......h
```



These encrypted DMG volumes can be mounted using the `hdiutil` command shown above.

If the password is unknown, John the Ripper and CrowbarDMG are able to access and attempt to brute force these volumes.

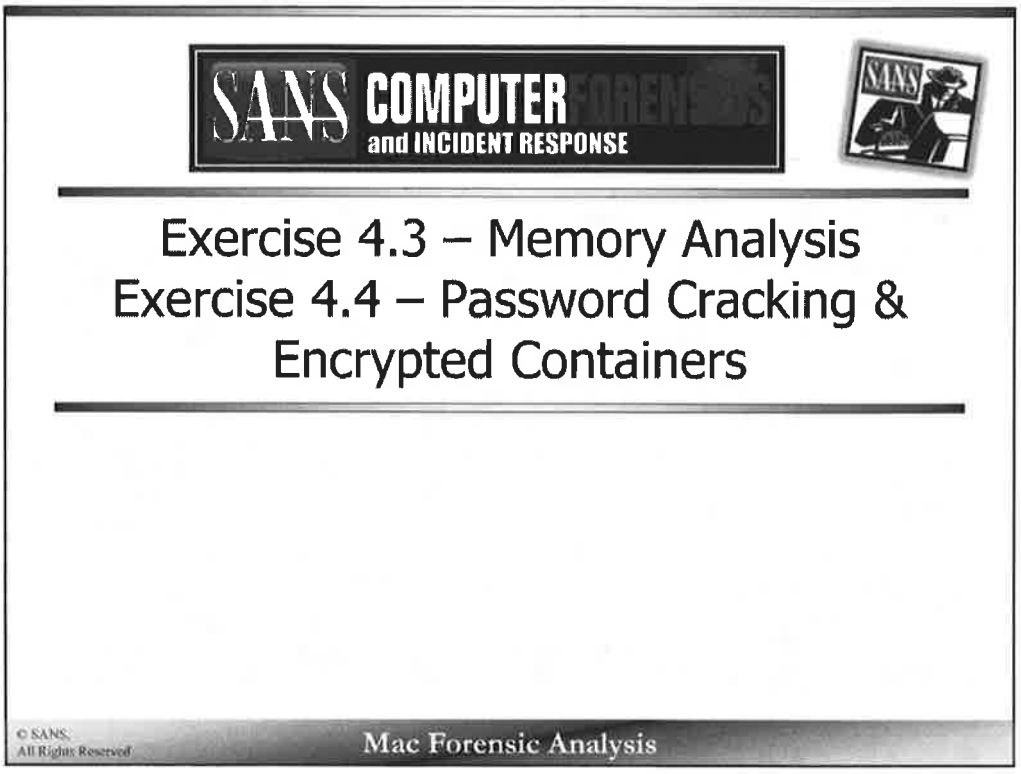
A special program called `dmg2john` will have to be used with John the Ripper to extract the hash. This utility is available in the Jumbo JTR release.



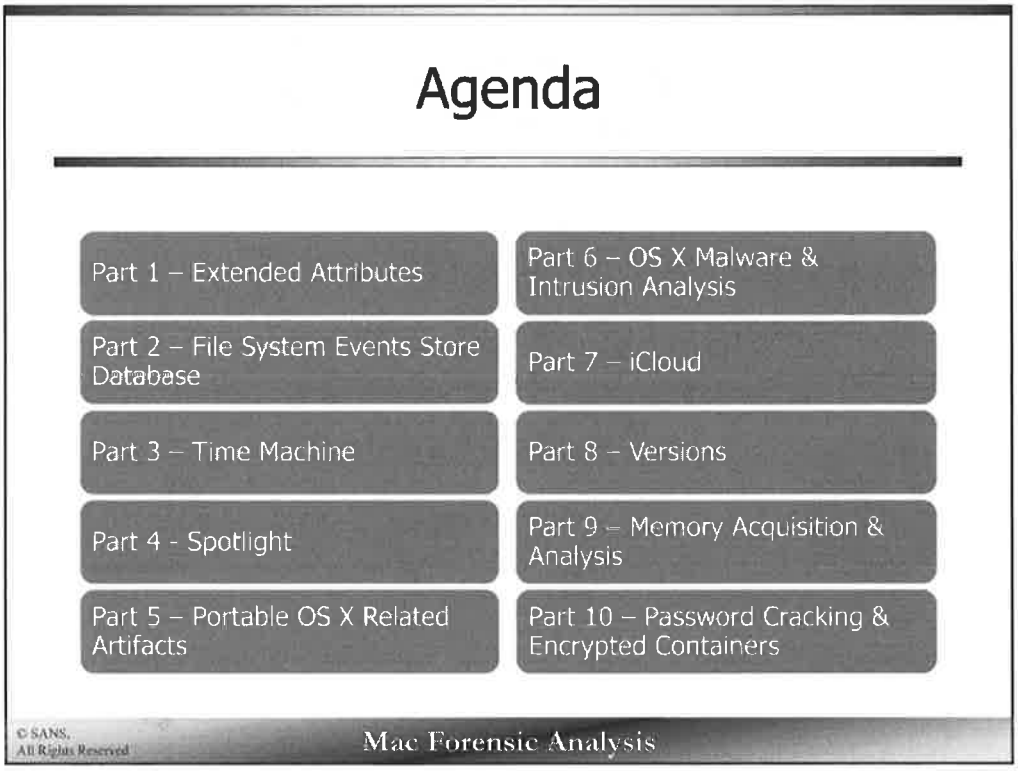
One way to make password brute-forcing faster is to use a dictionary file. If you were lucky enough to capture an RAM dump at the time of acquisition, very often plaintext passwords are stored in RAM.

To create a dictionary file, use the `strings` command, uniquely sort the output, and save the contents in a text file to be used with your password cracking software of choice.

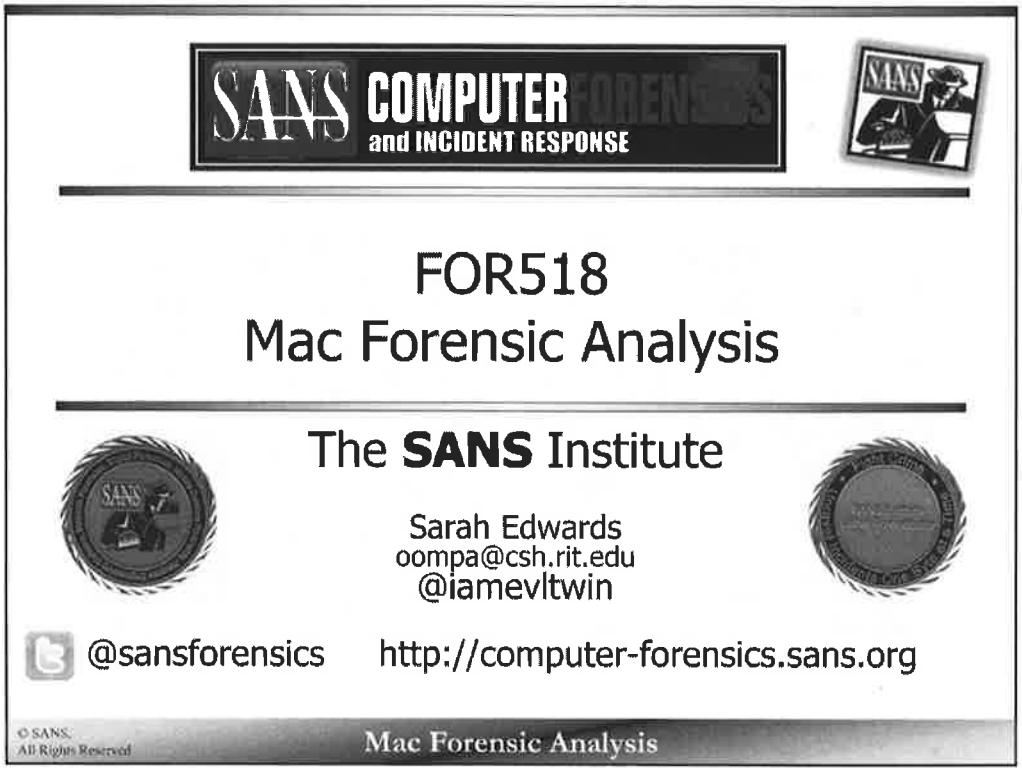
```
strings <MemoryImage> | sort -u > dictionary.txt
```



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