518.3

User Data, System Configuration, and Log Analysis

SANS | GIAC CERTIFICATIONS

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

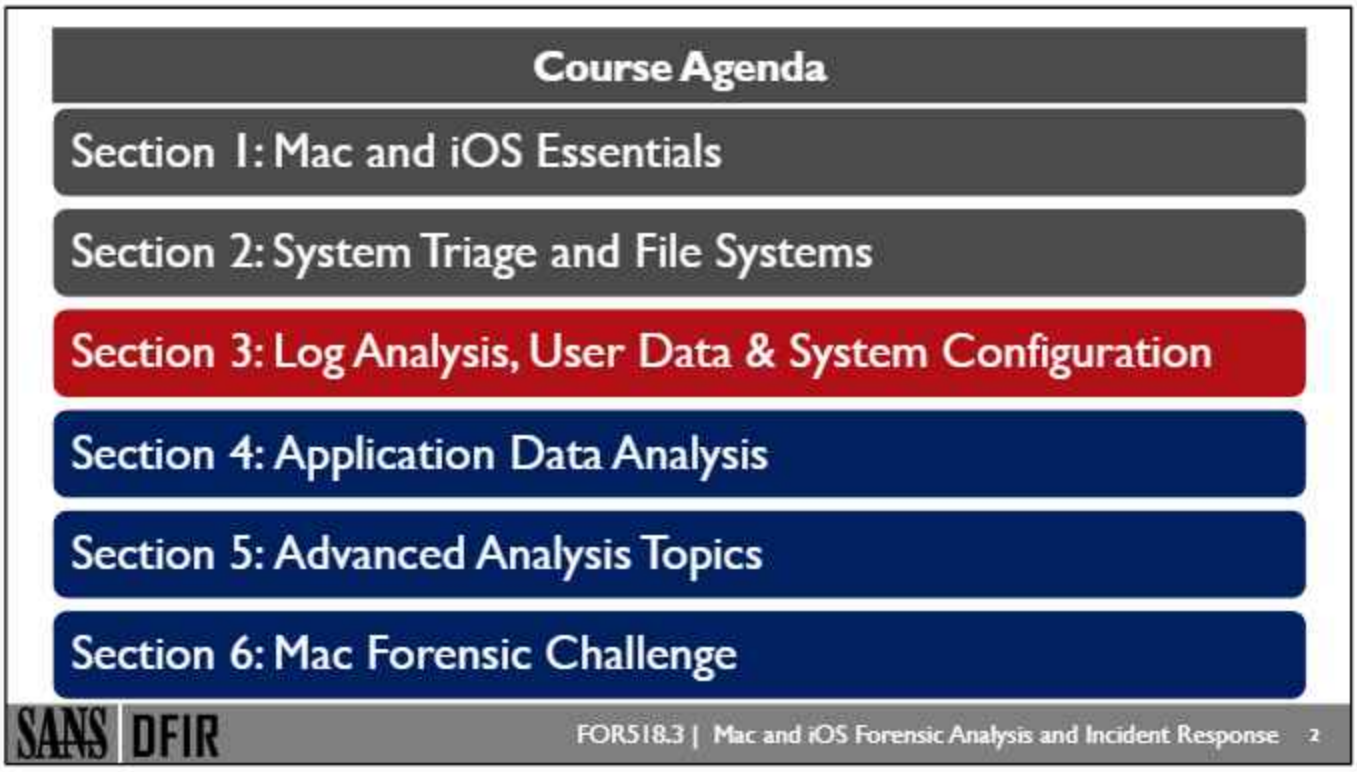
All reference links are operational in the browser-based delivery of the electronic workbook.

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

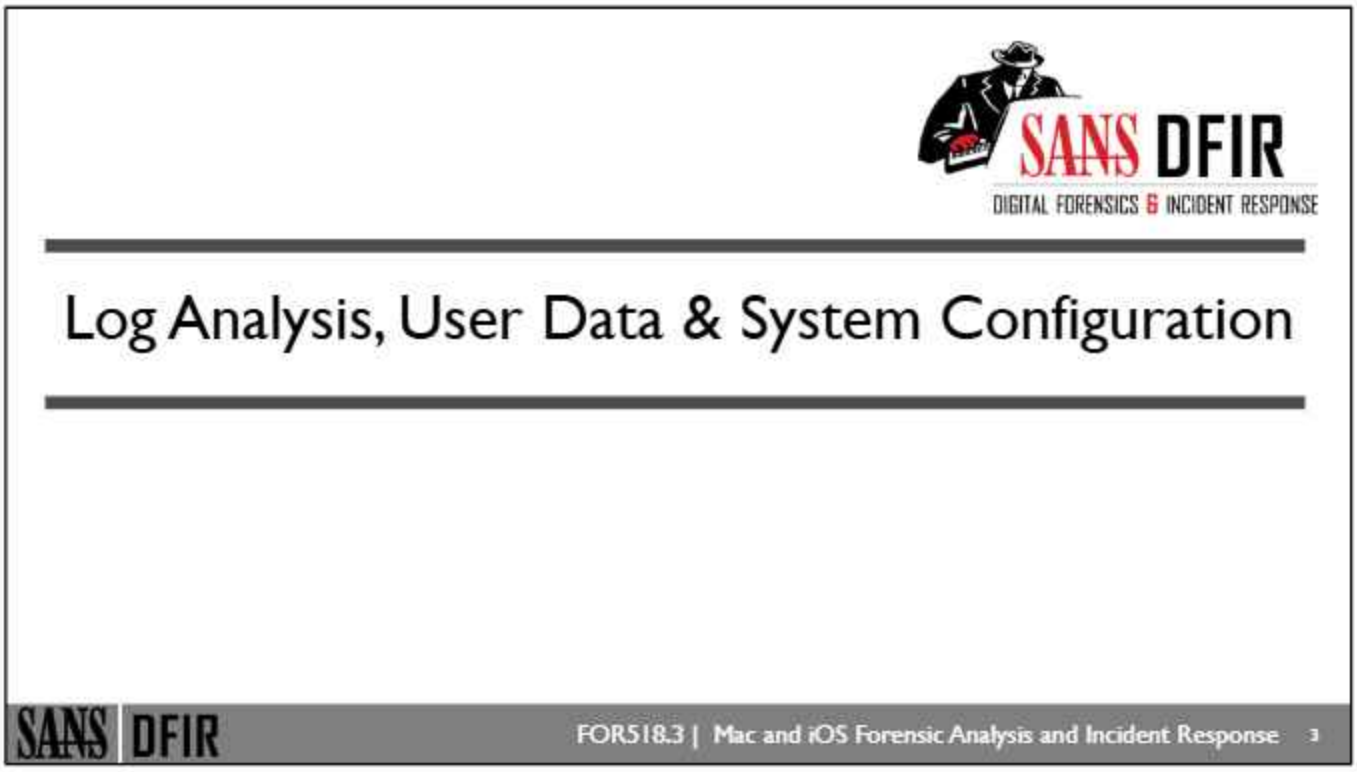
User Data, System Configuration, and Log Analysis

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

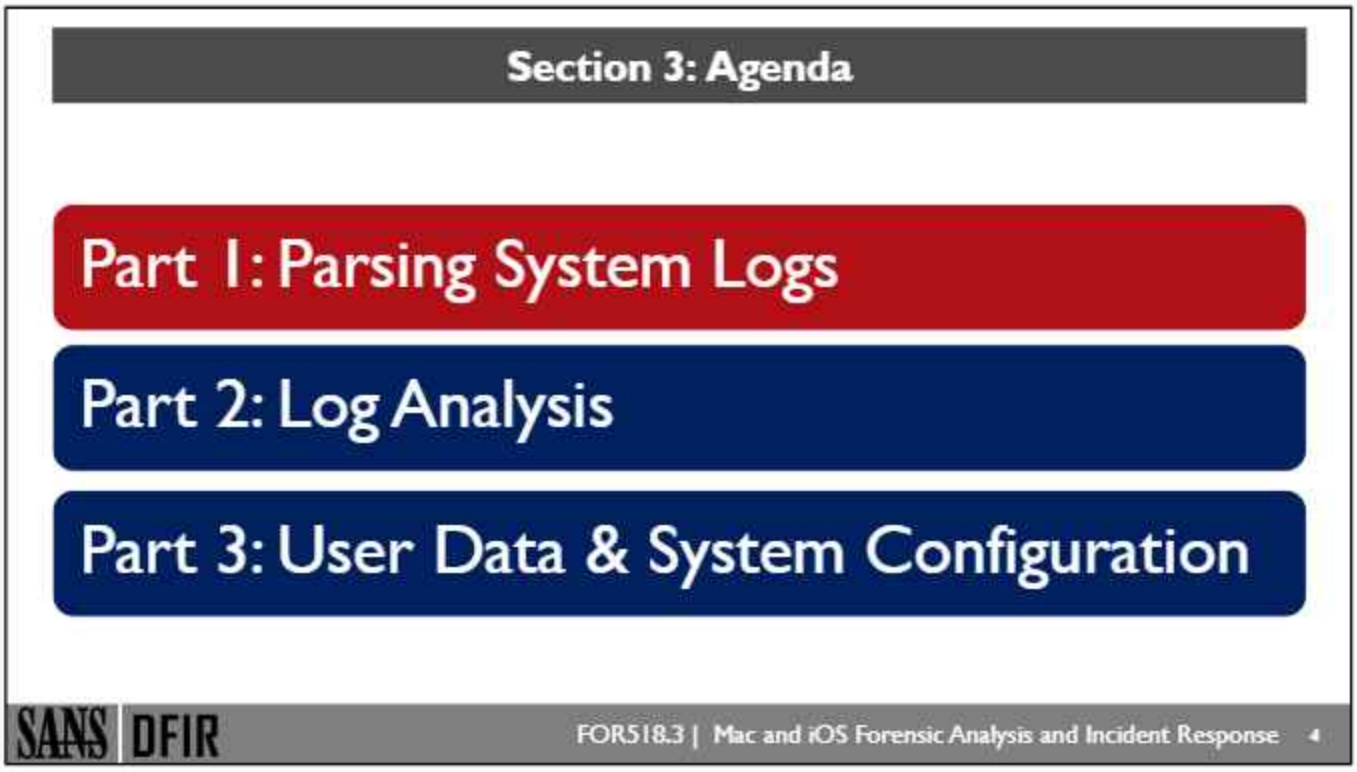
https://digital-forensics.sans.org/
https://twitter.com/sansforensics



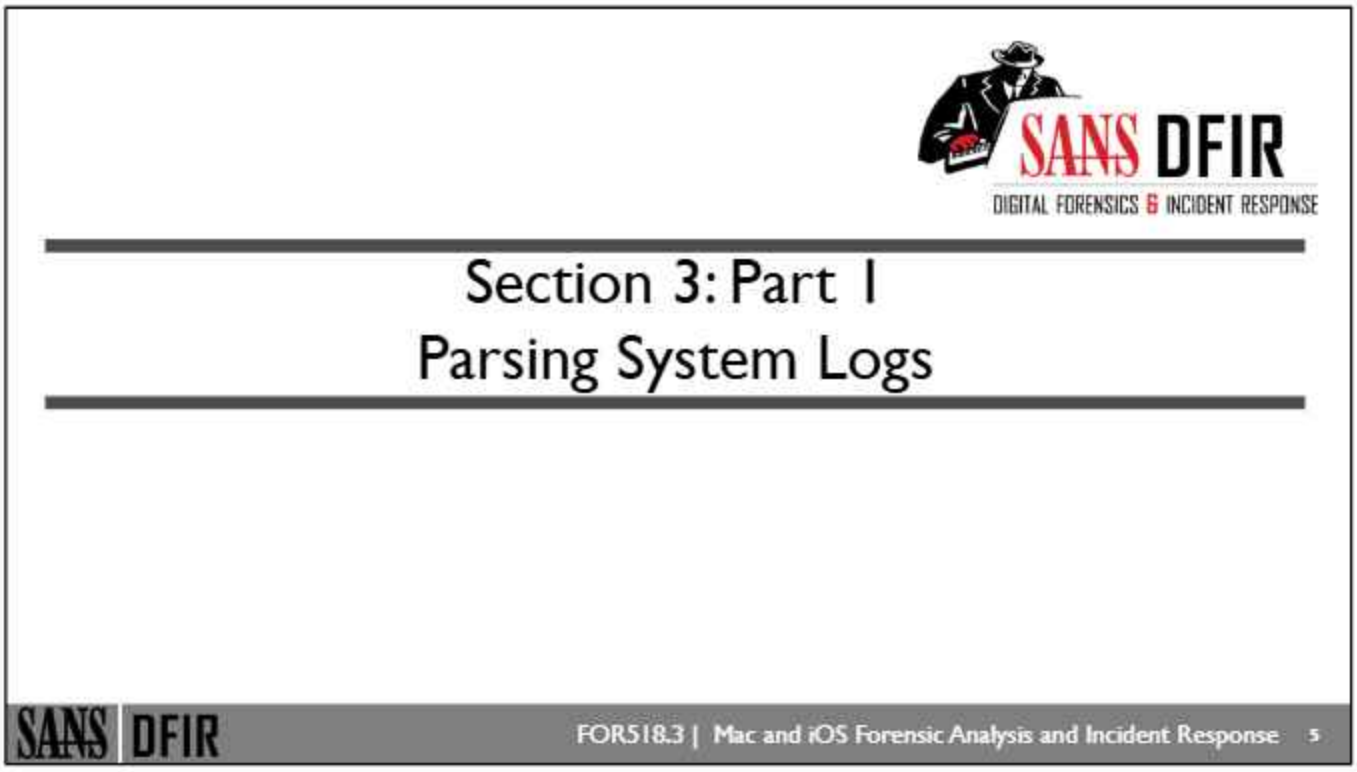
This page intentionally left blank.



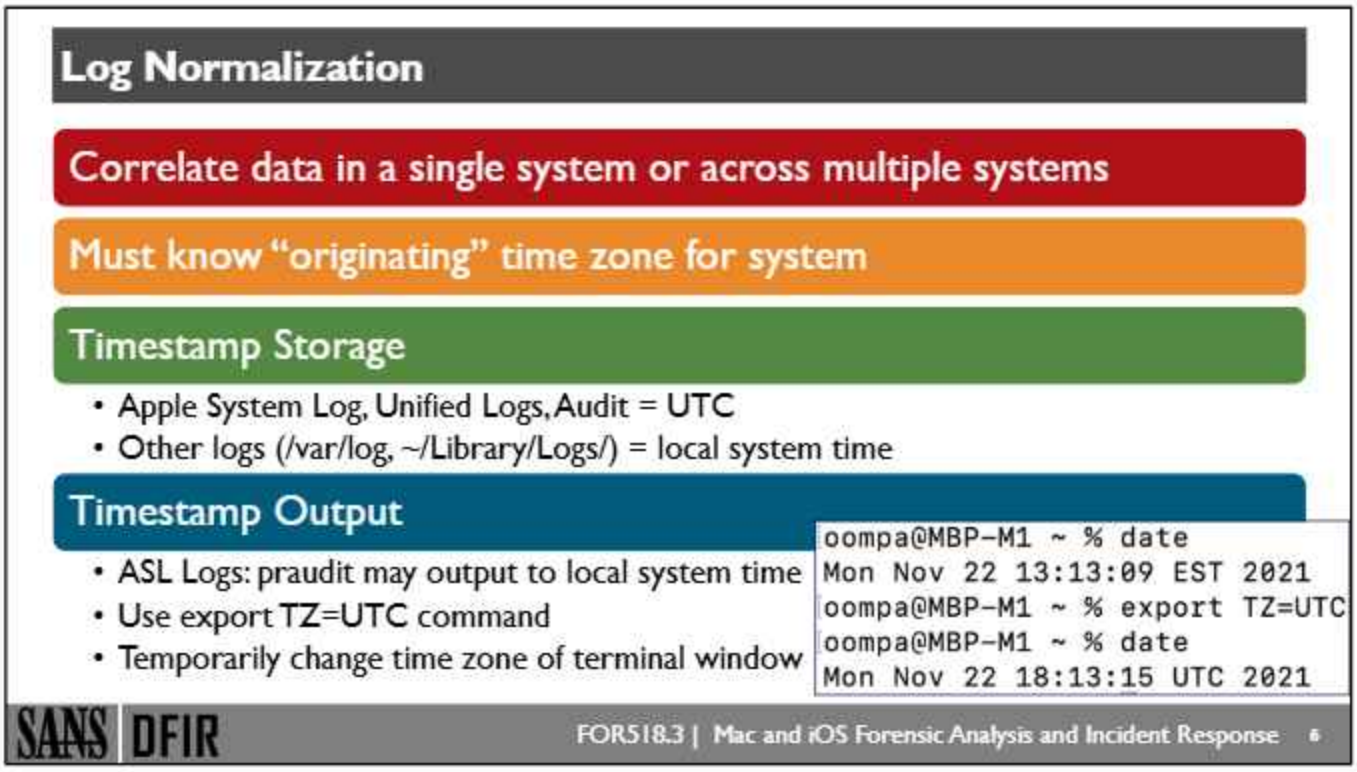
This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.



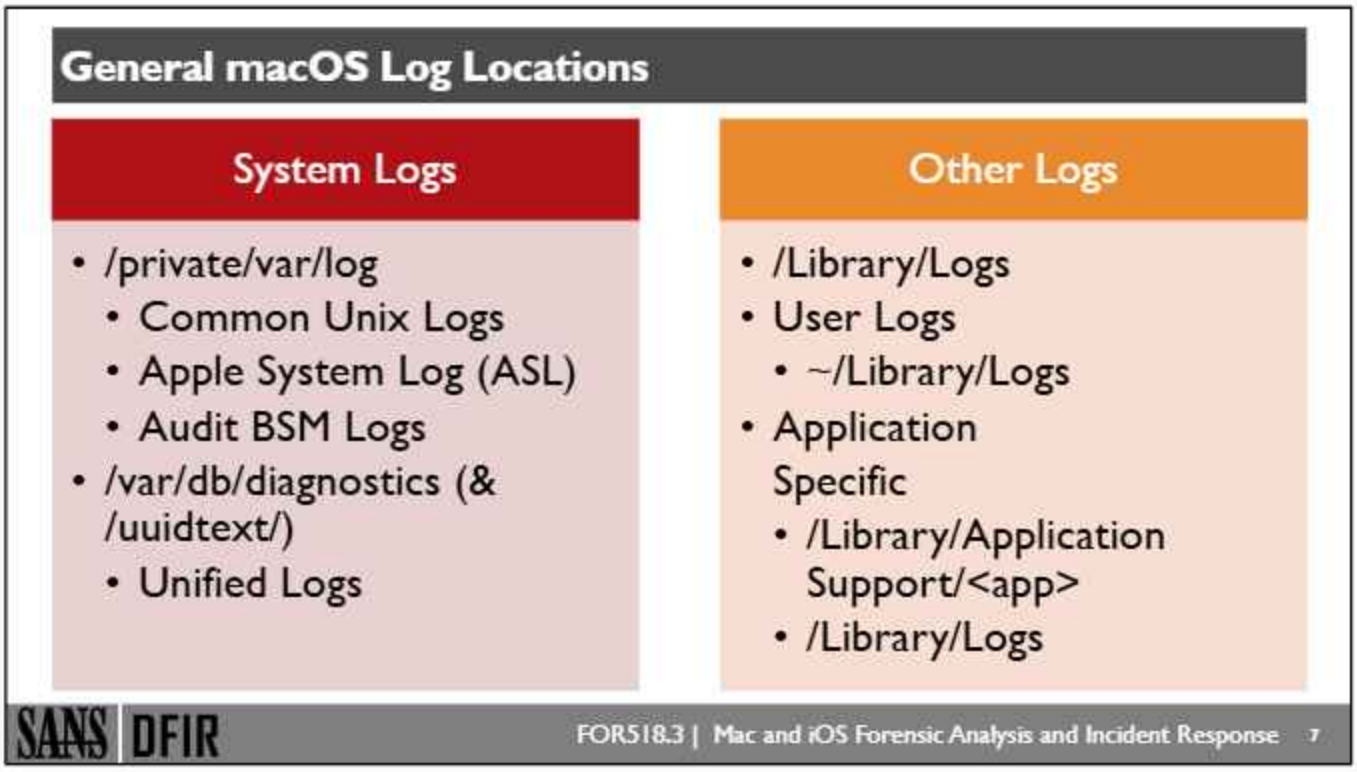
Analysts often have to correlate information across multiple systems across different time zones. To combine and understand what each system was doing at the same time, the analyst will need to normalize the log.

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

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

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

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



There are three primary locations in macOS where logs are found.

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

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

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

Common Unix Logs

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

```
ocmpa@MBP-M1 log % ls -l system.log*
-rw-r-----@ 1 root  admin  893793 Mar 14 14:31 system.log
-rw-r-----  1 root  admin   48027 Mar 14 00:00 system.log.0.gz
-rw-r-----  1 root  admin   64879 Mar 13 00:00 system.log.1.gz
-rw-r-----  1 root  admin   57563 Mar 12 00:00 system.log.2.gz
-rw-r-----  1 root  admin   69181 Mar 11 00:27 system.log.3.gz
-rw-r-----  1 root  admin   68821 Mar 10 00:00 system.log.4.gz
-rw-r-----  1 root  admin   38448 Mar  9 00:16 system.log.5.gz
-rw-r-----  1 root  admin   65560 Mar  8 00:00 system.log.6.gz
ocmpa@MBP-M1 log % head system.log
Mar 14 00:00:01 --- last message repeated 1 time ---
Mar 14 00:00:01 MBP-M1 syncdefaultsd[56365]: objc[56365]: Class SYOClient is implemented in both /System/Library/PrivateFrameworks/SyncedDefaults.framewor
k/Versions/A/SyncedDefaults and /System/Library/PrivateFrameworks/SyncedDefaults.framework/Support/syncdefaultsd. One of the two will be used. Which one i
s undefined.
Mar 14 00:00:01 MBP-M1 syncdefaultsd[56364]: objc[56364]: Class SYOClient is implemented in both /System/Library/PrivateFrameworks/SyncedDefaults.framewor
k/Versions/A/SyncedDefaults and /System/Library/PrivateFrameworks/SyncedDefaults.framework/Support/syncdefaultsd. One of the two will be used. Which one i
s undefined.
```


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

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

Bzip2 or Gzip Decompression

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

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


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

- 0 = newest
- 7 = oldest

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

Apple System Log

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

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


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

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

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

Reference:
`asl.conf` Man Page

Apple System Log: Filenames

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

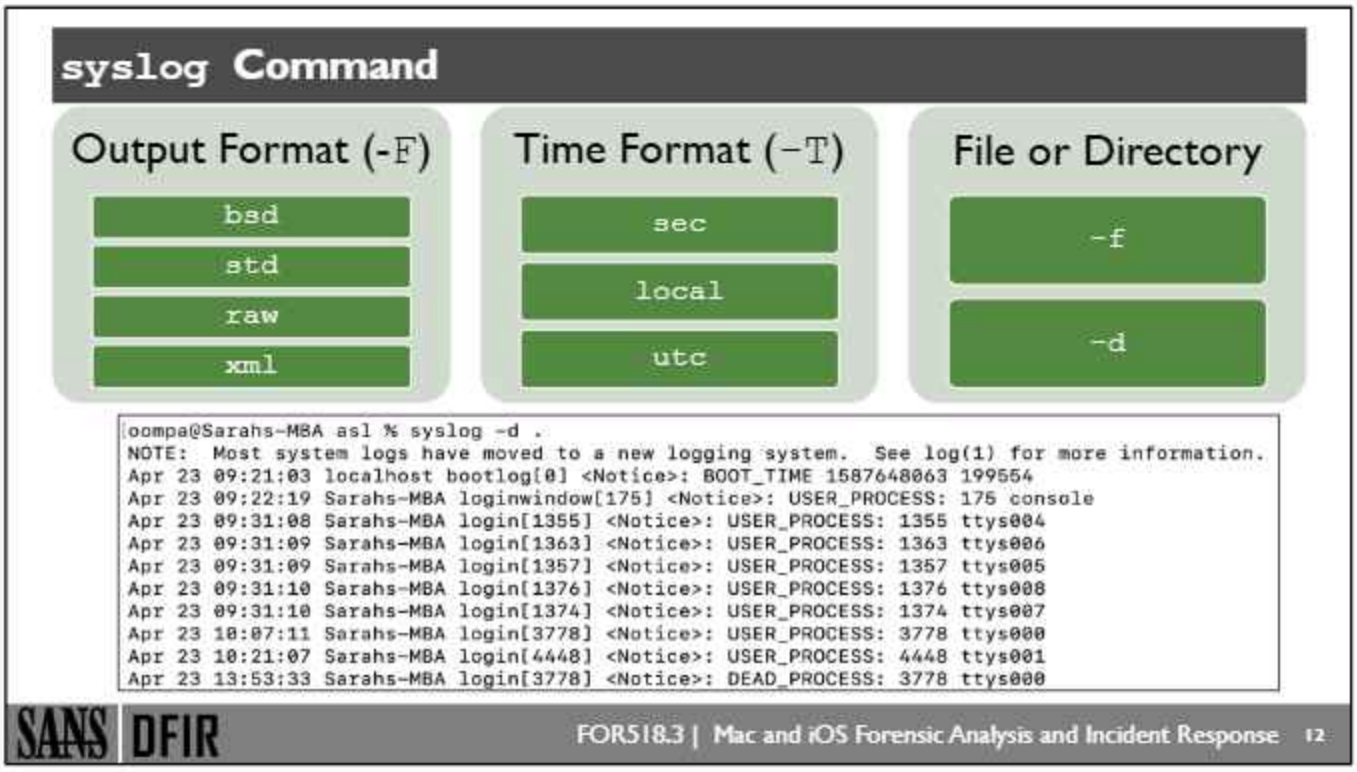
```
oompa@Sarahs-MBA AUX.2020.06.22 % pwd
/private/var/log/asl/AUX.2020.06.22
oompa@Sarahs-MBA AUX.2020.06.22 % file *
1648852: ASCII text
1648864: ASCII text
oompa@Sarahs-MBA AUX.2020.06.22 % cat 1645854
timestamp: 1592765964428
isAnonymous: true
deviceConfigId: 5612
investigationId: 0
model: "MacBookAir8,1"
softwareBuild: "19H287"
buildtype: "User"
tz_offset: -14400
metric_file_type: 1
metricLogs {
  triggerTime: 1592765961416
  triggerId: 3801088
  profileId: 212
  handoffUserActivityBecameCurrent {
    timestamp: 1592765961416
    bundleIdentifier: "unknown"
    activityType: "NSUserActivityTypeBrowsingWeb"
  }
}
```

```
oompa@Sarahs-MBA asl % pwd
/private/var/log/asl
oompa@Sarahs-MBA asl % ls -la
total 108984
drwxr-xr-x  23 root  wheel       736 Jun 22 12:02 .
drwxr-xr-x  52 root  wheel      1664 Jun 22 21:08 ..
-rw-------+  1 root  wheel   2246003 Jun 15 23:56 2020.06.15.G80.asl
-rw-------+  1 root  wheel   3448573 Jun 16 23:37 2020.06.16.G80.asl
-rw-------+  1 root  wheel   3828585 Jun 17 23:27 2020.06.17.G80.asl
-rw-------+  1 root  wheel   4856695 Jun 18 22:33 2020.06.18.G80.asl
-rw-------+  1 root  wheel   9124485 Jun 19 23:01 2020.06.19.G80.asl
-rw-------+  1 root  wheel   9475154 Jun 20 22:56 2020.06.20.G80.asl
-rw-------+  1 root  wheel  10283891 Jun 21 23:36 2020.06.21.G80.asl
-rw-------+  1 root  wheel   9324490 Jun 22 21:53 2020.06.22.G80.asl
drwxr-xr-x@  4 root  wheel       128 Jun 15 17:24 AUX.2020.06.15
drwxr-xr-x@  4 root  wheel       128 Jun 16 17:36 AUX.2020.06.16
drwxr-xr-x@  4 root  wheel       128 Jun 17 18:18 AUX.2020.06.17
drwxr-xr-x@  4 root  wheel       128 Jun 18 16:06 AUX.2020.06.18
drwxr-xr-x@  4 root  wheel       128 Jun 19 14:25 AUX.2020.06.19
drwxr-xr-x@  4 root  wheel       128 Jun 20 16:11 AUX.2020.06.20
drwxr-xr-x@  4 root  wheel       128 Jun 21 12:02 AUX.2020.06.21
drwxr-xr-x@  4 root  wheel       128 Jun 22 12:02 AUX.2020.06.22
-rw-------+  1 root  wheel     41984 Apr 27 18:44 BB.2021.04.30.G80.asl
-rw-------+  1 root  wheel     65148 May 30 16:36 BB.2021.05.31.G80.asl
-rw-------+  1 root  wheel     34715 Jun 22 21:52 BB.2021.06.30.G80.asl
drwxr-xr-x   8 root  wheel       256 Jun 22 21:08 Logs
-rw-r--r--@  1 root  wheel        12 Jun 22 21:53 StoreData
oompa@Sarahs-MBA asl % ls -l AUX.2020.06.22
total 464
-rw-r--r--@ 1 root  wheel  168937 Jun 22 12:02 1645852
-rw-r--r--@ 1 root  wheel   64421 Jun 22 12:02 1645854
```


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

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

The directories starting with AUX or "Auxiliary" are used on 10.8+ systems. These directories contain text files comprised of additional data that is referenced by the `syslog` file.



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

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

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

The time output can also be formatted:

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

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

syslog -T utc -F raw -d /asl

- [ASLMessageID 1]
- [Time 2020-04-23 13:21:03Z]
- [TimeNanoSec 199554000]
- [Level 5]
- [PID 0]
- [UID 0]
- [GID 0]
- [ReadGID 80]
- [Host localhost]
- [Sender bootlog]
- [Facility com.apple.system.utmpx]
- [Message BOOT_TIME 1587648063 199554]
- [ut_id 0x00 0x00 0x00 0x00]
- [ut_pid 1]
- [ut_type 2]
- [ut_tv.tv_sec 1587648063]
- [ut_tv.tv_usec 199554]
- [ASLExpireTime 1619270463]

```
oompa@Sarahs-MBA asl % syslog -T utc -F raw -d .
NOTE:  Most system logs have moved to a new logging system.  See log(1) for more information.
[ASLMessageID 1] [Time 2020-04-23 13:21:03Z] [TimeNanoSec 199554000] [Level 5] [PID 0] [UID 0] [GID 0] [ReadGID 80] [Host localhost] [Sender bootlog] [Faci
lity com.apple.system.utmpx] [Message BOOT_TIME 1587648063 199554] [ut_id 0x00 0x00 0x00 0x00] [ut_pid 1] [ut_type 2] [ut_tv.tv_sec 1587648063] [ut_tv.tv_u
sec 199554] [ASLExpireTime 1619270463]
[ASLMessageID 212] [Time 2020-04-23 13:22:19Z] [TimeNanoSec 684877000] [Level 5] [PID 175] [UID 0] [GID 0] [ReadGID 80] [Host Sarahs-MBA] [Sender loginwind
ow] [Facility com.apple.system.lastlog] [Message USER_PROCESS: 175 console] [ut_user oompa] [ut_id 0x2f 0x00 0x01 0x01] [ut_line console] [ut_pid 175] [ut_
type 7] [ut_tv.tv_sec 1587648139] [ut_tv.tv_usec 684543] [SenderMachUUID 869C4C29-42B8-3BB5-8ECD-F15EF6B8F5D8] [ASLExpireTime 1619270539]
```


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

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

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

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

```
oompa@Sarahs-MBA asl % syslog -T utc -F raw -d .
NOTE:  Most system logs have moved to a new logging system.  See log(1) for more information.
[ASLMessageID 1] [Time 2020-04-23 13:21:03Z] [TimeNanoSec 199554000] [Level 5] [PID 0] [UID 0] [GID 0] [ReadGID 80] [Host localhost] [Sender bootlog] [Faci
lity com.apple.system.utmpx] [Message BOOT_TIME 1587648063 199554] [ut_id 0x00 0x00 0x00 0x00] [ut_pid 1] [ut_type 2] [ut_tv.tv_sec 1587648063] [ut_tv.tv_u
sec 199554] [ASLExpireTime 1619270463]
[ASLMessageID 212] [Time 2020-04-23 13:22:19Z] [TimeNanoSec 684877000] [Level 5] [PID 175] [UID 0] [GID 0] [ReadGID 80] [Host Sarahs-MBA] [Sender loginwind
ow] [Facility com.apple.system.lastlog] [Message USER_PROCESS: 175 console] [ut_user oompa] [ut_id 0x2f 0x00 0x01 0x01] [ut_line console] [ut_pid 175] [ut_
type 7] [ut_tv.tv_sec 1587648139] [ut_tv.tv_usec 684543] [SenderMachUUID 869C4C29-42BB-3BB5-8ECD-F15EF6B8FBD8] [ASLExpireTime 1619270539]
```

Major Log Changes in 10.12 and iOS 10: Unified Logging

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

Supersedes Apple System Logs (ASL) and Syslog

macOS and iOS (Physical) Locations:

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

Binary Files: *.tracev3

```
Sarahs-MacBook-Pro:diagnostics oompa$ sw_vers
ProductName:    Mac OS X
ProductVersion: 10.13.1
BuildVersion:   17B1003
Sarahs-MacBook-Pro:diagnostics oompa$ pwd
/var/db/diagnostics
Sarahs-MacBook-Pro:diagnostics oompa$ ls -la
total 2848
drwxr-x---   10 root  admin      320 Dec 15 21:38 .
drwxr-xr-x   85 root  wheel     2720 Dec 16 03:46 ..
drwxr-xr-x    2 root  admin       64 Nov 11 13:17 HighVolume
drwxr-xr-x   55 root  admin     1760 Dec 16 13:19 Persist
drwxr-xr-x  407 root  admin    13024 Dec 16 17:09 Special
-rw-r--r--    1 root  admin   265220 Dec 16 17:11 logdata.statistics.0.txt
-rw-r--r--    1 root  admin  1049501 Dec 12 19:58 logdata.statistics.1.txt
-rw-r--r--    1 root  admin   125296 Dec 15 13:23 shutdown.log
drwxr-xr-x    3 root  wheel       96 Nov 11 13:19 timesync
-rw-r--r--    1 root  admin      484 Dec 15 21:38 version.plist
Sarahs-MacBook-Pro:diagnostics oompa$ ls -l Persist/
total 1058008
-rw-r--r--@ 1 root  admin  10489048 Dec  4 19:25 0000000000000095.tracev3
-rw-r--r--@ 1 root  admin  10483632 Dec  4 19:46 0000000000000096.tracev3
-rw-r--r--@ 1 root  admin  10483608 Dec  4 20:08 0000000000000097.tracev3
-rw-r--r--@ 1 root  admin  10487128 Dec  4 20:33 0000000000000098.tracev3
-rw-r--r--@ 1 root  admin  10482456 Dec  4 20:56 0000000000000099.tracev3
-rw-r--r--@ 1 root  admin  10490336 Dec  4 21:17 000000000000009a.tracev3
-rw-r--r--@ 1 root  admin  10484864 Dec  4 21:40 000000000000009b.tracev3
-rw-r--r--@ 1 root  admin  10482976 Dec  4 22:02 000000000000009c.tracev3
```

SANS DFIR | FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 16

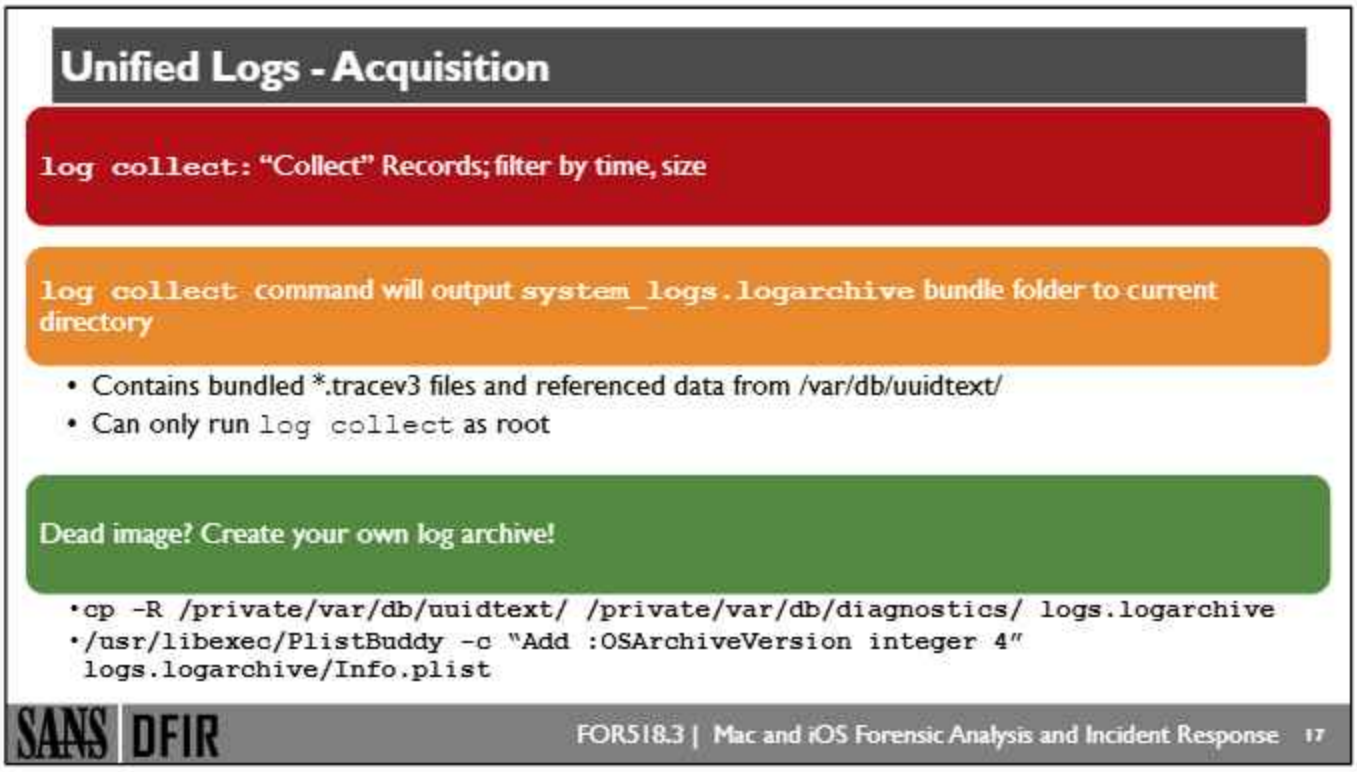
Introduced with macOS 10.12 and iOS 10 is unified logging, which is supposed to be uniform across devices and operating systems. This directory has changed over many point versions since it was released in 10.12. The screenshot above is from a 10.13.1 system, while it used to appear like the screenshot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

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

The main log files are stored in a binary format.

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

```
bash-3.2# pwd
/private/var/db/diagnostics
bash-3.2# ls -l
total 206272
drwxr-xr-x    2 root  wheel        68 Sep 27 19:03 Events
drwxr-xr-x   38 root  wheel      1820 Oct 22 21:52 FaultsAndErrors
drwxr-xr-x    2 root  wheel        68 Sep 27 19:03 Oversize
drwxr-xr-x    2 root  wheel        68 Sep 27 19:03 SpecialHandling
drwxr-xr-x    3 root  wheel        68 Sep 27 19:03 StateDumps
drwxr-xr-x   14 root  wheel       476 Oct 22 22:03 TTL
-rw-r-----    1 root  wheel   3428816 Oct 14 20:27 logdata.Persistent.20161014T005422.tracev3
-rw-r--r--    1 root  wheel   9167544 Oct 15 13:11 logdata.Persistent.20161015T002740.tracev3
-rw-r-----    1 root  wheel  10491576 Oct 15 21:18 logdata.Persistent.20161015T171217.tracev3
-rw-r-----    1 root  wheel   3818584 Oct 17 19:22 logdata.Persistent.20161016T012225.tracev3
-rw-r-----    1 root  wheel   5676560 Oct 17 20:38 logdata.Persistent.20161017T232211.tracev3
-rw-r-----    1 root  wheel   6168968 Oct 18 20:20 logdata.Persistent.20161018T004411.tracev3
-rw-r--r--    1 root  wheel   8171240 Oct 19 21:02 logdata.Persistent.20161019T004806.tracev3
-rw-r-----    1 root  wheel   9050872 Oct 21 19:38 logdata.Persistent.20161020T010211.tracev3
-rw-r-----    1 root  wheel  10497776 Oct 22 15:05 logdata.Persistent.20161022T000721.tracev3
-rw-r--r--    1 root  wheel   5683528 Oct 22 21:52 logdata.Persistent.20161022T190750.tracev3
-rw-r-----    1 root  wheel  10537456 Oct 23 12:08 logdata.Persistent.20161023T010231.tracev3
-rw-r-----    1 root  wheel  10493240 Oct 23 13:21 logdata.Persistent.20161023T161202.tracev3
-rw-r-----    1 root  wheel  10567792 Oct 23 16:41 logdata.Persistent.20161023T172230.tracev3
-rw-r-----    1 root  wheel    983232 Oct 23 17:19 logdata.Persistent.20161023T204633.tracev3
-rw-r--r--    1 root  wheel    718216 Oct 23 16:46 logdata.statistics.0.txt
-rw-r--r--    1 root  wheel    384313 Oct 15 13:11 shutdown.log
```



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

This command can only be run with root privileges.

The logarchive bundle requires an Info.plist to be created with a key 'OSArchiveVersion" set to '4', when log show is run. If this is not done, the manually created logarchive file will not be able to be read with the proper timestamps. More details here: https://www.cellebrite.com/en/converting-unified-logs-a-great-disturbance-in-the-force/

References:
Man Page for `log` Command
https://developer.apple.com/documentation/os/logging
https://developer.apple.com/videos/play/wwdc2016/721/
https://www.cellebrite.com/en/converting-unified-logs-a-great-disturbance-in-the-force/

Parse Unified Log Files

Import into Console.app

- Streaming view within Console.app shows your log messages. Some messages may be volatile

`log show`: "Show" Records

- Filter on predicates, time frame, etc.
- `log show`: On live system, output system logs to standard out, already in temporal order

Perform on *.logarchive, *.tracev3 (file or directory) extracted from image

- By File: `*.tracev3`: Must be within log archive.
- By Log Archive: `*.logarchive`: In temporal order

By default, disregards Info and Debug messages

- Use `--info, --debug, --signpost` for all log messages

Standard Output to Terminal

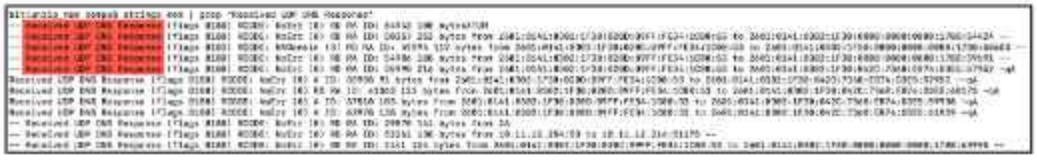
- `--style` = (JSON or Syslog Outputs)

SANS DFIR | FOR518.3 | Mac and iOS Forensic Analysis and Incident Response | 18

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

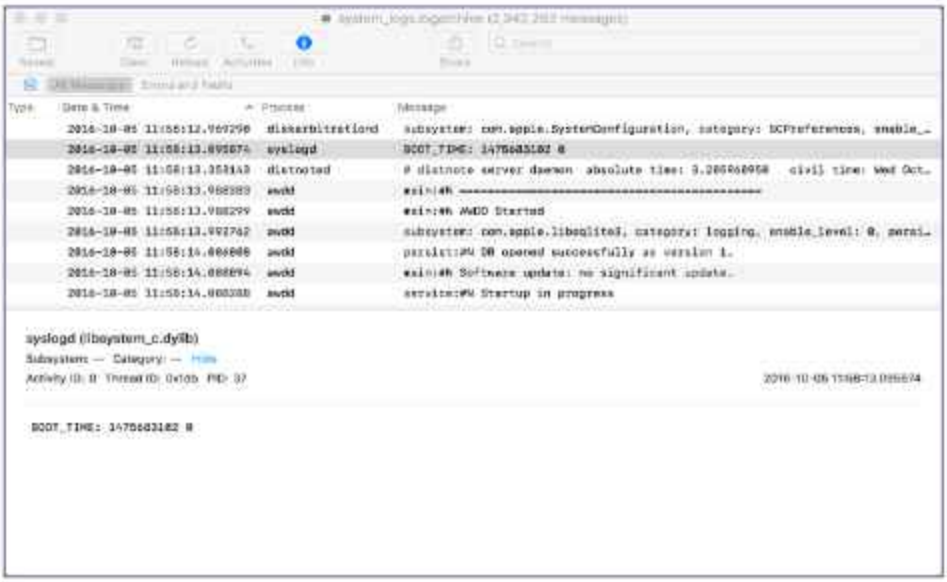
With Unified Logs, some messages are only stored in memory. The example below shows a "`Volatile`" column in the Console.app application, where the message is marked volatile or not (1 = Volatile, 0 = Non-volatile). Looking for the same messages on a "dead" system may not show you all the messages you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

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



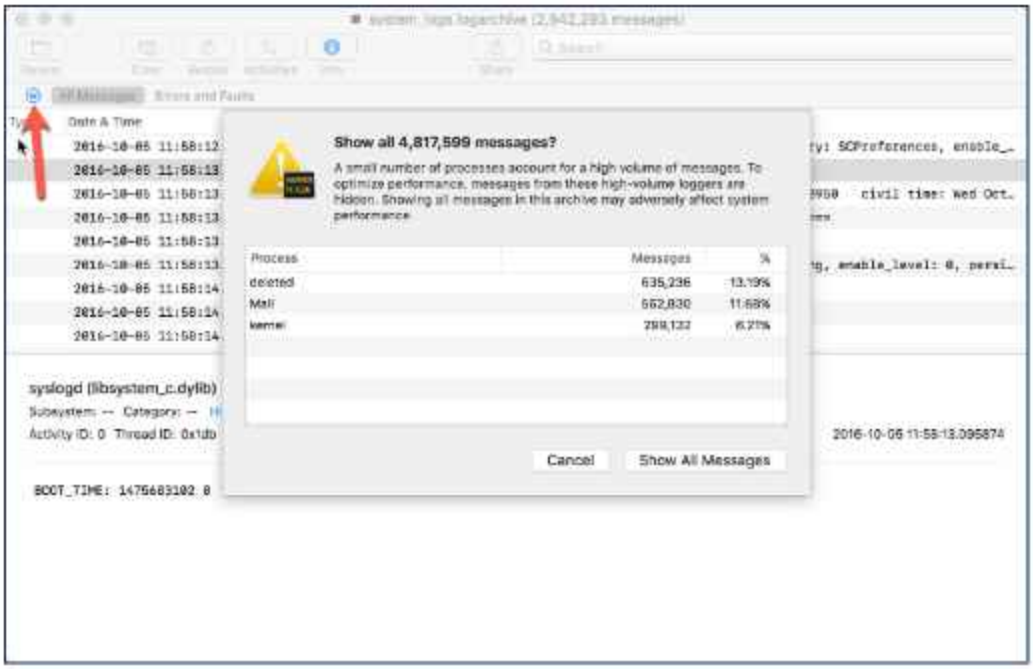
Message	Volatile	Subsystem
— Received UDP DNS Response (flags 8180) RCODE: NoErr (0) RD RA ID: 11920 69 bytes from 2601:0141:.	1	com.apple.mDNS…
1 Questions	1	com.apple.mDNS…
0 d.dropbox.com. Addr	1	com.apple.mDNS…
3 Answers	1	com.apple.mDNS…
0 TTL 76 17 d.dropbox.com. CNAME d.v.dropbox.com.	1	com.apple.mDNS…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

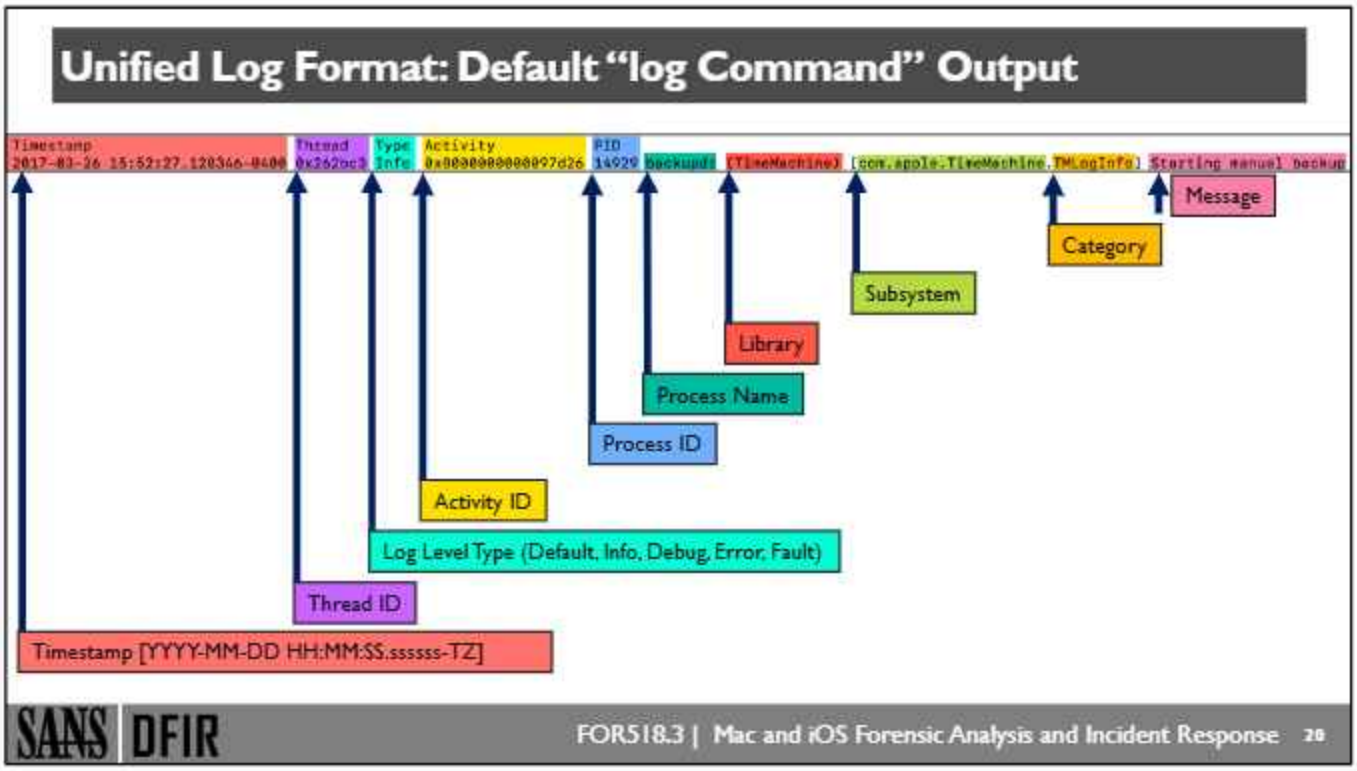
Parse Unified Log Files: `Console.app`



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

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





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

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

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

Filter log events using logical statements (AND/OR/NOT/</>/=/CONTAINS/BEGINSWITH/TRUE/FALSE, etc.)

- **eventType**: Filters on Event Types (Log, Trace, Activity)
- **eventMessage**: Filters on message text or Activity Name (if log/trace event)
- **messageType**: Filters on Log Level (Default, Info, Debug, Error, Fault)
- **processImagePath**: Filters on the process name of the event originator
- **senderImagePath**: Filters on sender name of the event originator (Library, framework, kext, mach-o binary)
- **subsystem**: Filters on subsystem (reverse DNS ID)
- **category**: Filters on subsystem category

[cd] = Ignore case/diacritics

```
Bit:LogsUnite compa$ log show system_logs.logarchive/ --info  --predicate 'processImagePath contains[cd] "backupd" and category contains[cd] "TMLogInfo" and eventMessage
 contains "Starting"' --start "2017-03-25 10:00:00"
==========
/Users/pomps/Dropbox (Personal)/LogsUnite/system_logs.logarchive
==========
Skipping debug messages, pass --debug to include.
Filtering the log data using "processImagePath CONTAINS[cd] "backupd" AND category CONTAINS[cd] "TMLogInfo" AND eventMessage CONTAINS "Starting""
Timestamp                       Thread     Type        Activity             PID
2017-03-25 15:09:52.159122-0400 0xbb011    Info        0x0                  9802   backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting automatic backup
2017-03-25 15:12:37.464085-0400 0xbb011    Info        0x0                  9802   backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting post-backup thinning
2017-03-25 16:21:36.193812-0400 0x187c14   Info        0x0                  12334  backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting automatic backup
2017-03-25 16:26:59.918274-0400 0x187e14   Info        0x0                  12334  backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting post-backup thinning
2017-03-25 17:21:32.389421-0400 0x21ab5c   Info        0x0                  12566  backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting automatic backup
2017-03-25 17:26:55.642558-0400 0x21ab5c   Info        0x0                  12566  backupd: (TimeMachine) [com.apple.TimeMachine.TMLogInfo] Starting post-backup thinning

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

SANS DFIR | FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 21

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

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

macOS Audit Logs - /private/var/audit/

Basic Security Module (BSM) Audit Logs

Binary Format

Filenames:

- Audit Trail Files: StartTime.EndTime Filenames - YYYYMMDDHHMMSS.YYYYMMDDHHMM
- "current" and *.not_terminated
- *.crash_recovery

```
-r--r-----  1 root  wheel     1146 Nov 13 13:06 20211113130630.20211113130633
-r--r-----  1 root  wheel    29928 Nov 13 13:07 20211113130633.20211113130710
-r--r-----  1 root  wheel   139603 Nov 13 17:47 20211113130710.crash_recovery
-r--r-----  1 root  wheel   213092 Nov 16 07:34 20211113175033.crash_recovery
-r--r-----  1 root  wheel    93916 Nov 17 07:17 20211116073637.crash_recovery
-r--r-----  1 root  wheel    65155 Nov 18 07:18 20211117072723.crash_recovery
-r--r-----  1 root  wheel    86392 Nov 19 07:14 20211118072006.crash_recovery
-r--r-----  1 root  wheel    52727 Nov 20 08:06 20211119071942.crash_recovery
-r--r-----  1 root  wheel    76317 Nov 20 18:24 20211120080700.crash_recovery
-r--r-----  1 root  wheel    32526 Nov 20 18:36 20211120182909.crash_recovery
-r--r-----  1 root  wheel    83519 Nov 21 17:16 20211120184048.crash_recovery
-r--r-----  1 root  wheel    89949 Nov 22 18:22 20211121171719.not_terminated
lrwxr-xr-x  1 root  wheel       40 Nov 21 17:17 current -> /var/audit/20211121171719.not_terminated
```

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

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

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

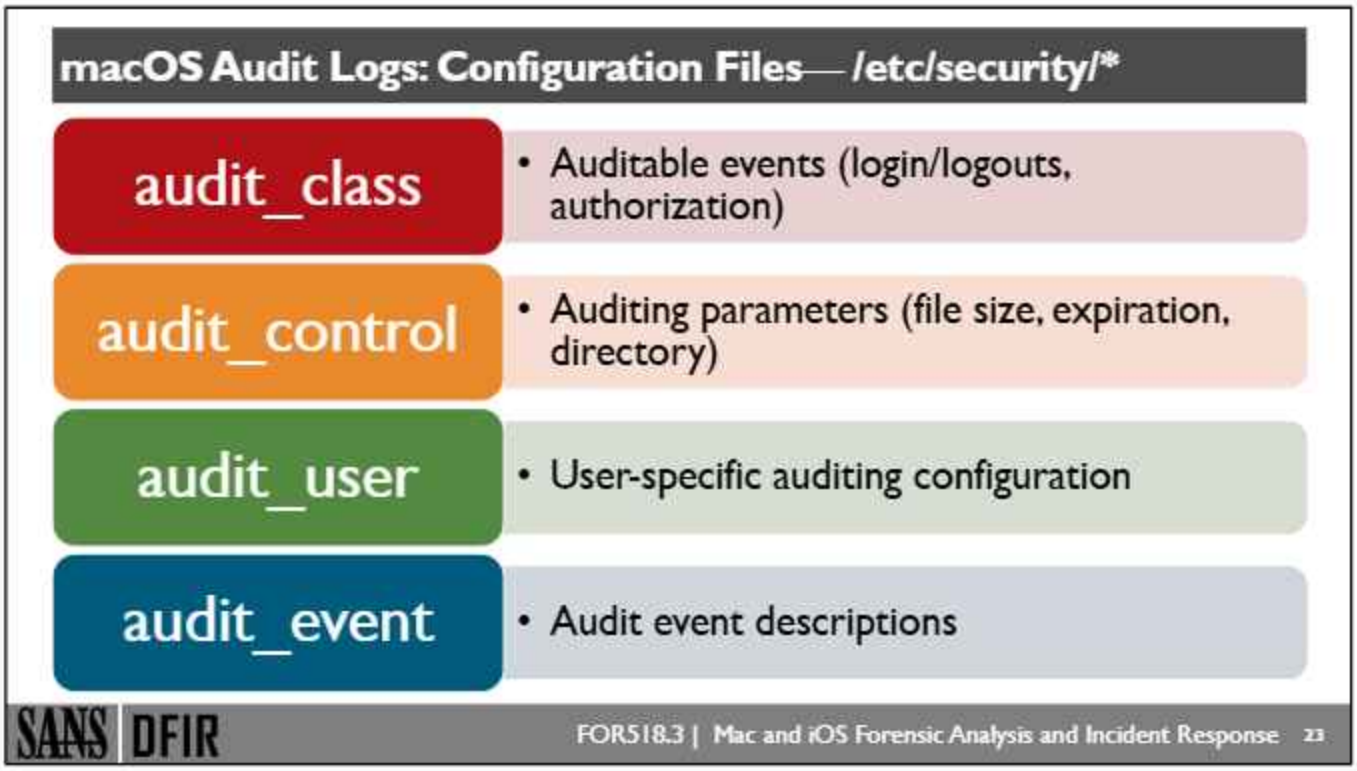
More information about BSM Audit logs can be found in Hal Pomeranz's article "Solaris Basic Security Mode (BSM) Auditing": www.deer-run.com/~hal/sysadmin/SolarisBSMAuditing.html.

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

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

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

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



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

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

This file contains different parameters.

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

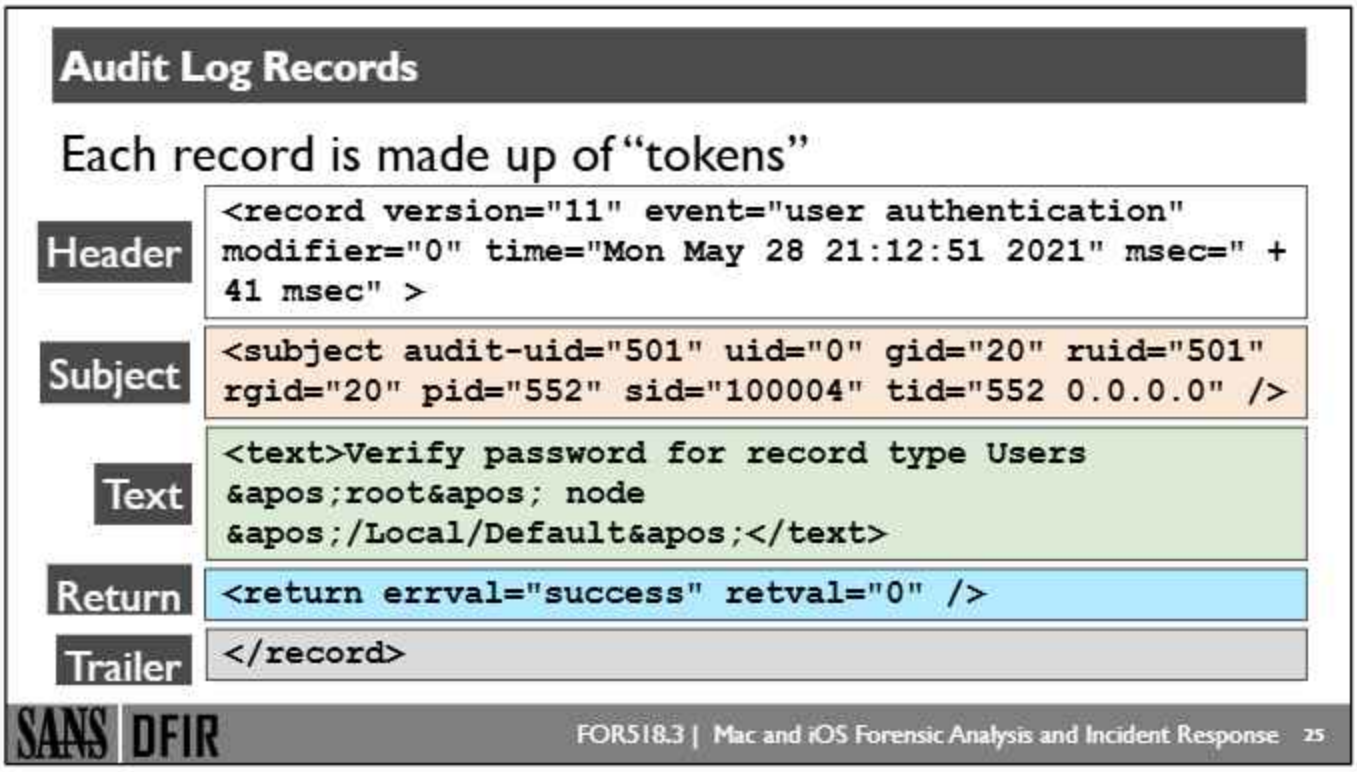
audit_user:
Each user can have specific audit functions recorded. The default configuration only includes one user—root.

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

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

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

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



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

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

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

More information about each token can be found on the `audit.log` Man Page.

Example from the Man Page for the Return and `Subject` tokens.

Reference:
`audit.log` Man Page

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

```
<record version="11" event="user authentication" modifier="0" time="Mon
May 28 21:12:51 2021" msec=" + 41 msec" >
<subject audit-uid="501" uid="0" gid="20" ruid="501" rgid="20" pid="552"
sid="100004" tid="552 0.0.0.0" />
<text>Verify password for record type Users &apos;root&apos; node
&apos;/Local/Default&apos;</text>
<return errval="success" retval="0" />
</record>

<record version="11" event="user authentication" modifier="0" time="Mon
May 28 21:12:55 2021" msec=" + 449 msec" >
<subject audit-uid="501" uid="0" gid="20" ruid="501" rgid="20" pid="554"
sid="100004" tid="554 0.0.0.0" />
<text>Verify password for record type Users &apos;root&apos; node
&apos;/Local/Default&apos;</text>
<return errval="failure: Unknown error: 255" retval="5000" />
</record>
```


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

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

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

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

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

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

auditreduce Command

Filter audit records:

- Timestamp
- User
- Group
- Event Type (shown below, found in `/etc/security/audit_event`)
- More!

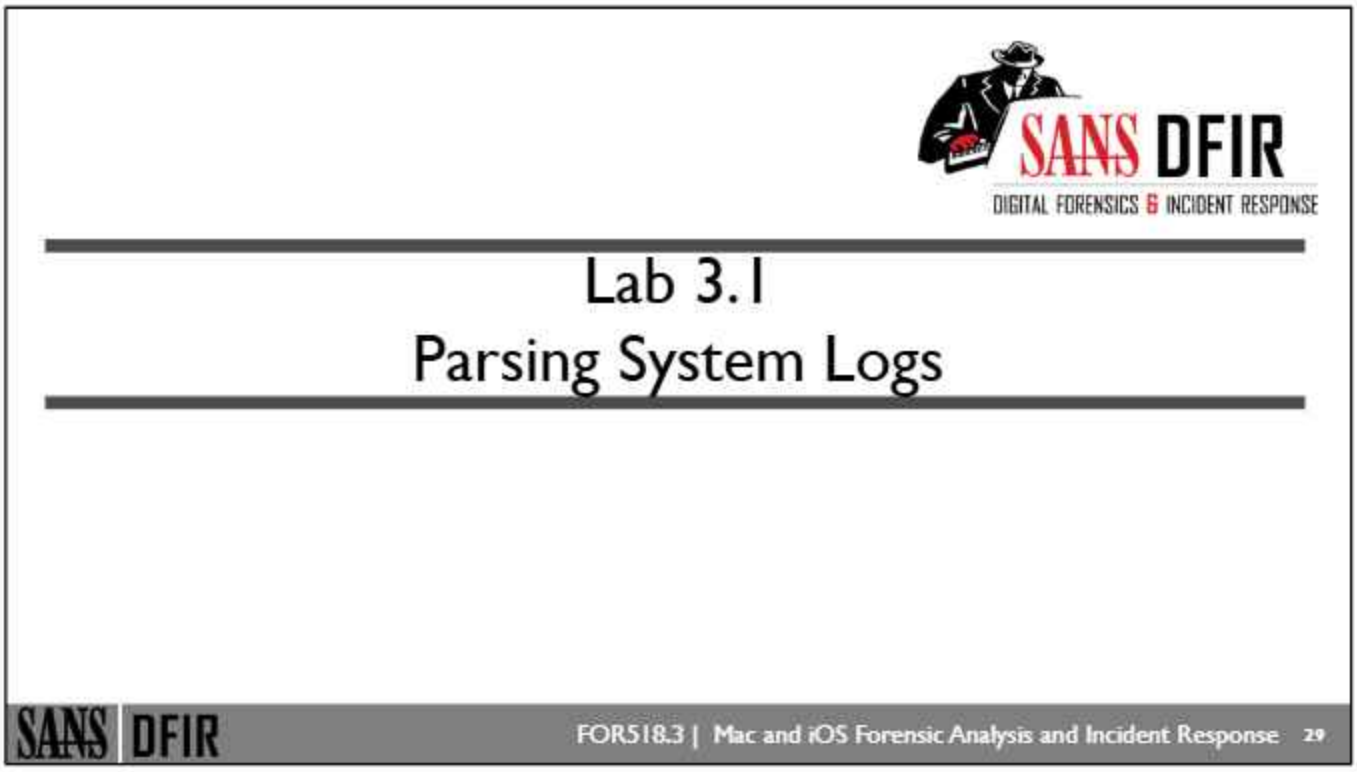
```
root@MBP-M1 audit # auditreduce -m AUE_lw_login 20220624121208.not_terminated | praudit -xn
<?xml version='1.0' encoding='UTF-8'?>
<audit>
<record version="11" event="loginwindow login" modifier="0" time="Fri Jun 24 08:12:14 2022" msec=" + 972 msec" >
<subject audit-uid="501" uid="0" gid="0" ruid="501" rgid="20" pid="396" sid="100018" tid="503316500.0.0.0" />
<return errval="success" retval="0" />
<identity signer-type="1" signing-id="com.apple.loginwindow" signing-id-truncated="no" team-id="" team-id-truncated="no"
cdhash="0x526c7ce4d91181492ee62c8bb6f2ffc1a8783a0f" />
</record>
```


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

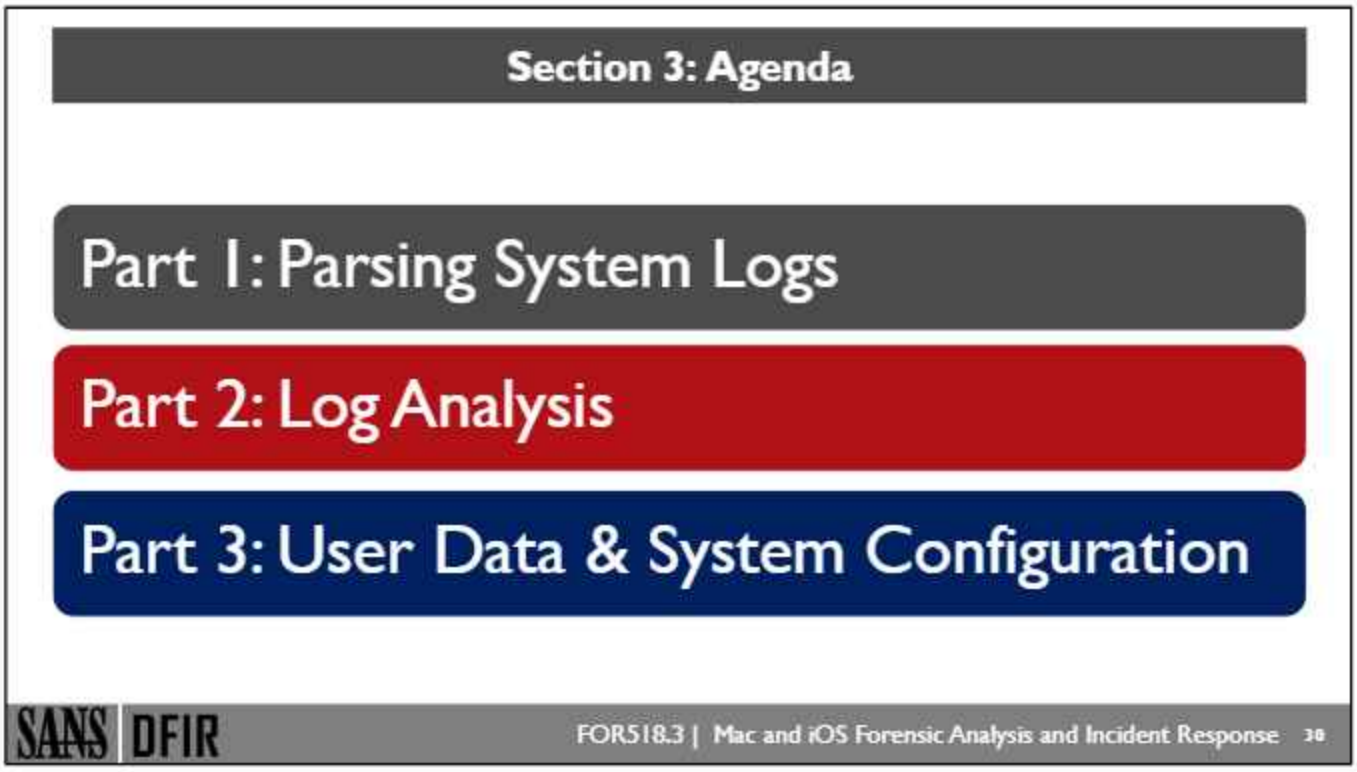
The screenshot shows a filter for a specific audit event to find login window logins (`AUE_lw_login`). The `auditreduce` command can be piped to the `praudit` command for easier viewing.

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

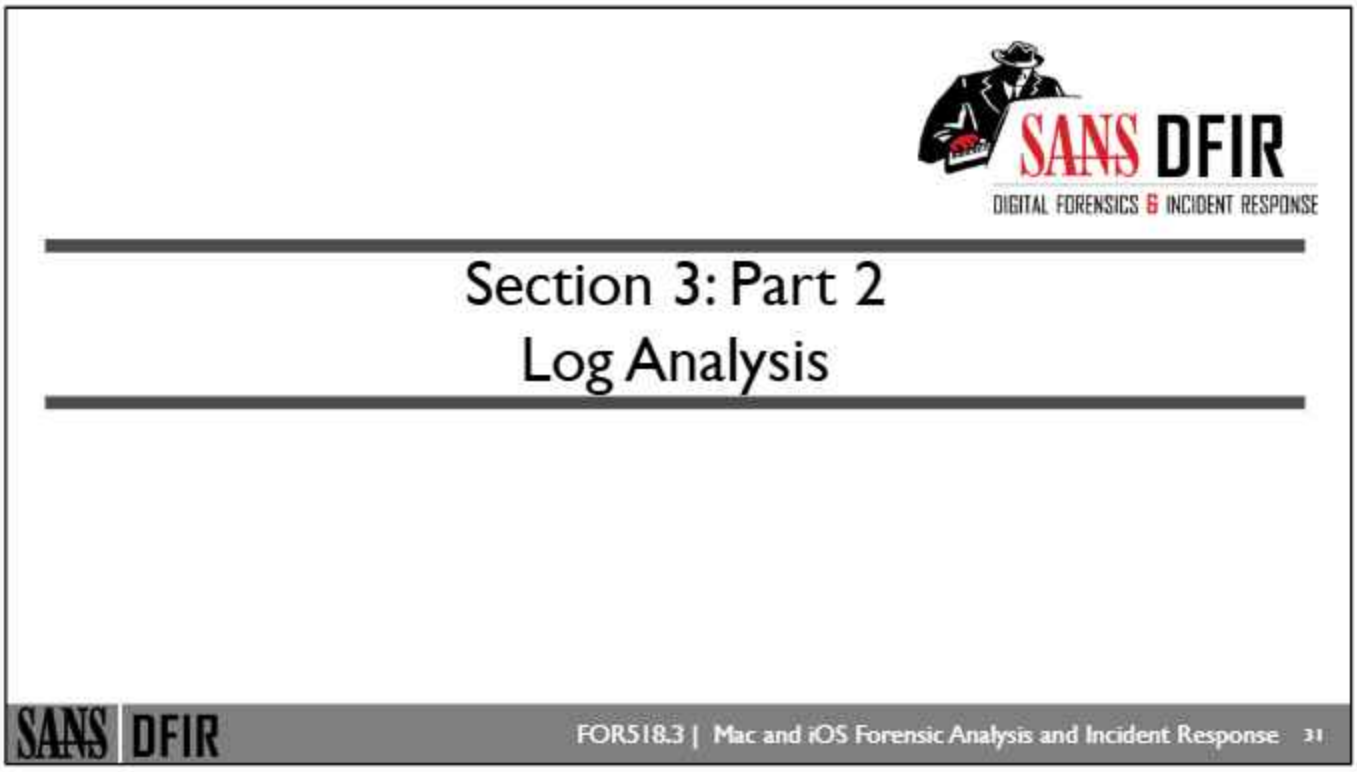
Reference:
`auditreduce` Man Page



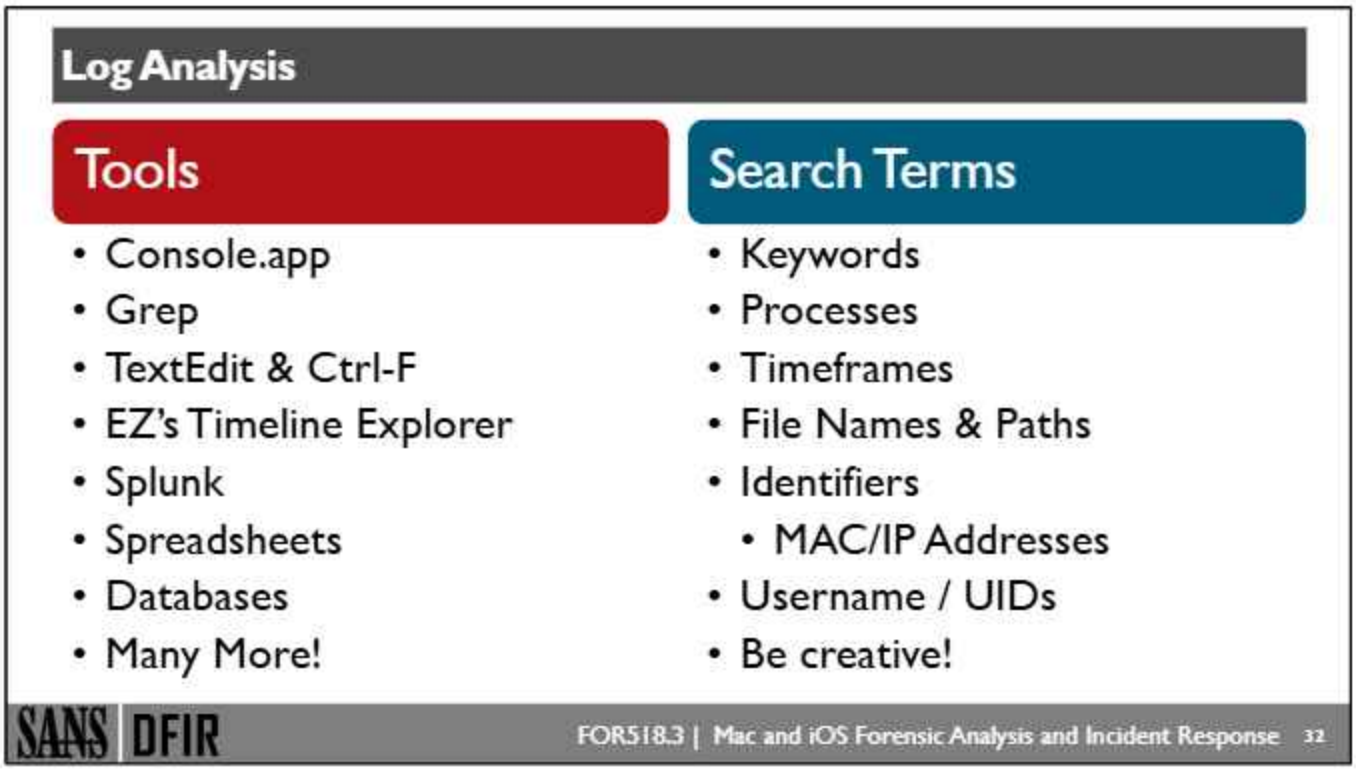
This page intentionally left blank.



This page intentionally left blank.

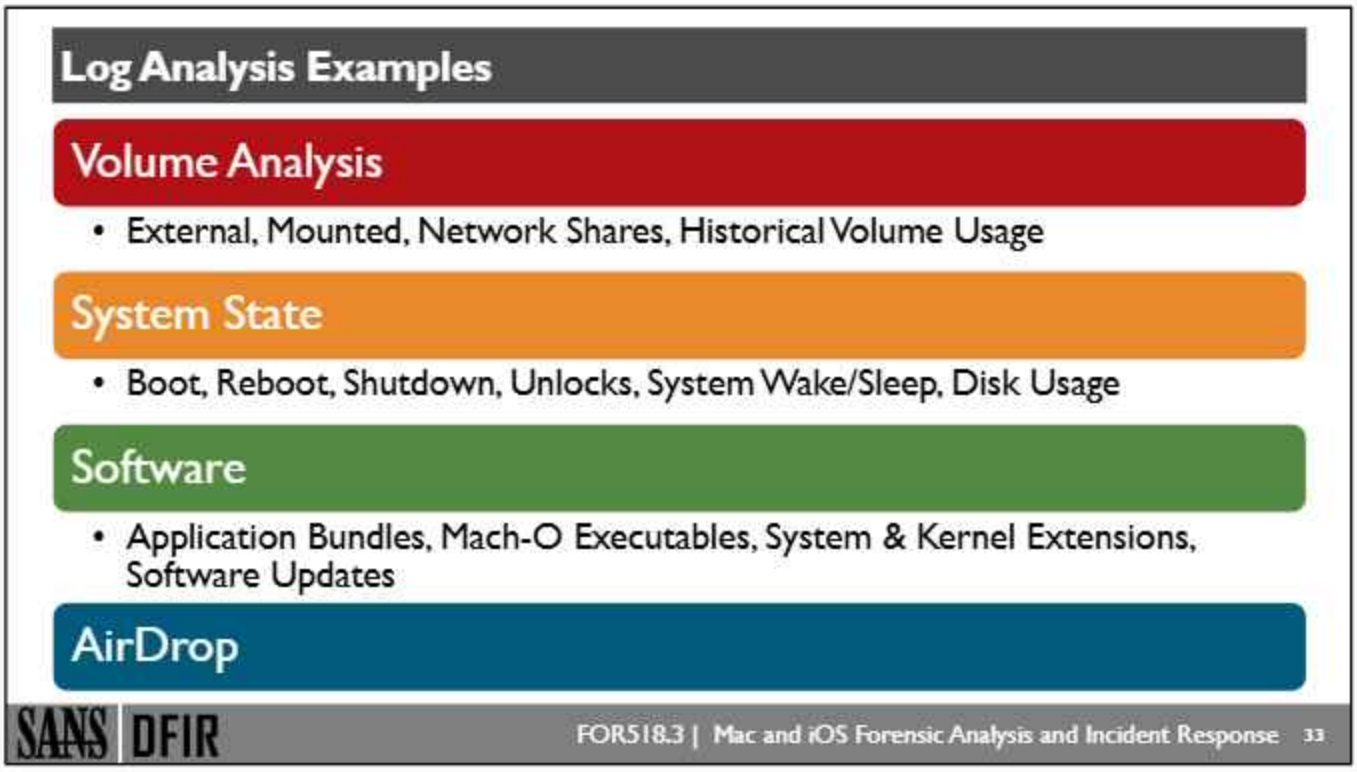


This page intentionally left blank.

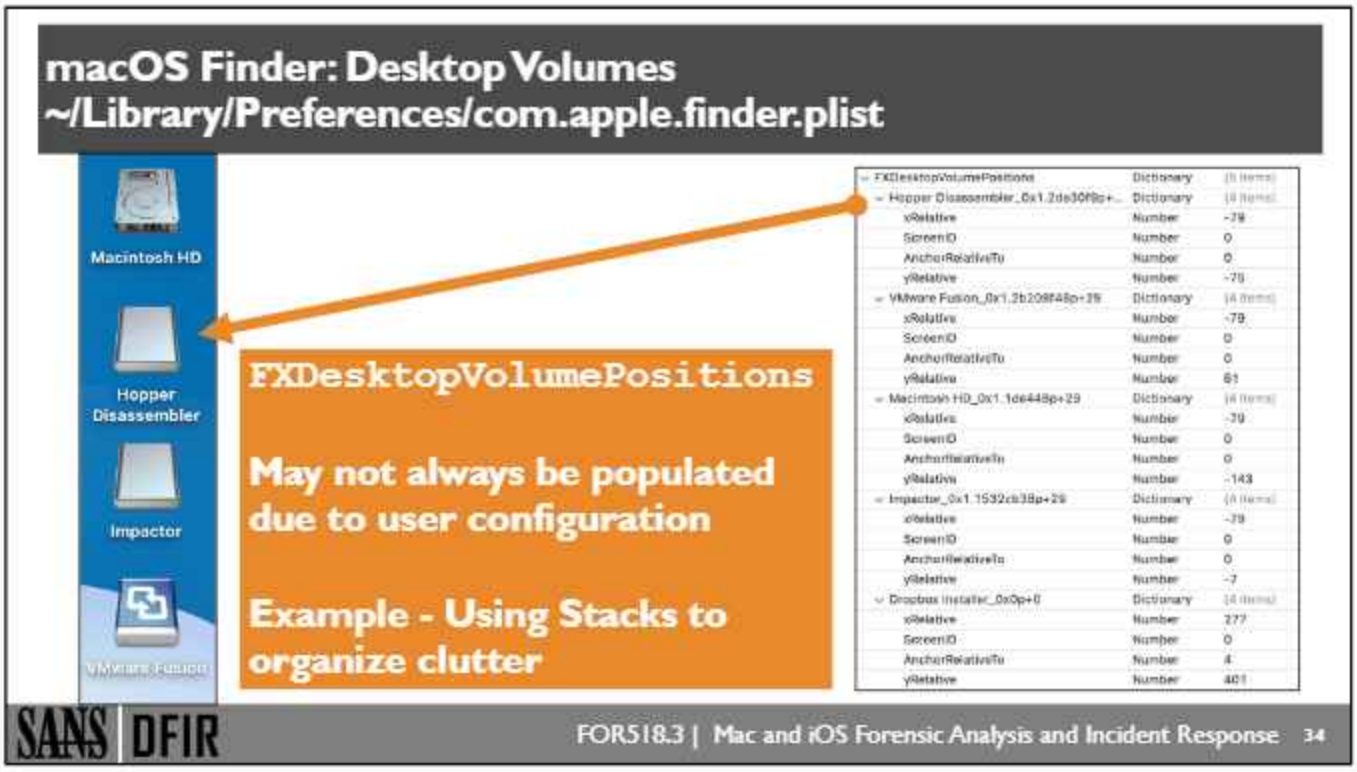


Creativity is the best method for log analysis, in this class we are going to focus on using Mac and *nix based tools to review our logs. However if you have a tool or method you prefer, please use those as well!

There can be many millions of log records, most folks will not likely have the time to spend scrolling through and will instead need to use specific search terms to find what they need. Again, creativity here is your best way to find what may be of interest to you in your investigations.



This page intentionally left blank.



The Finder application stores the mounted volumes in the `com.apple.finder.plist`.

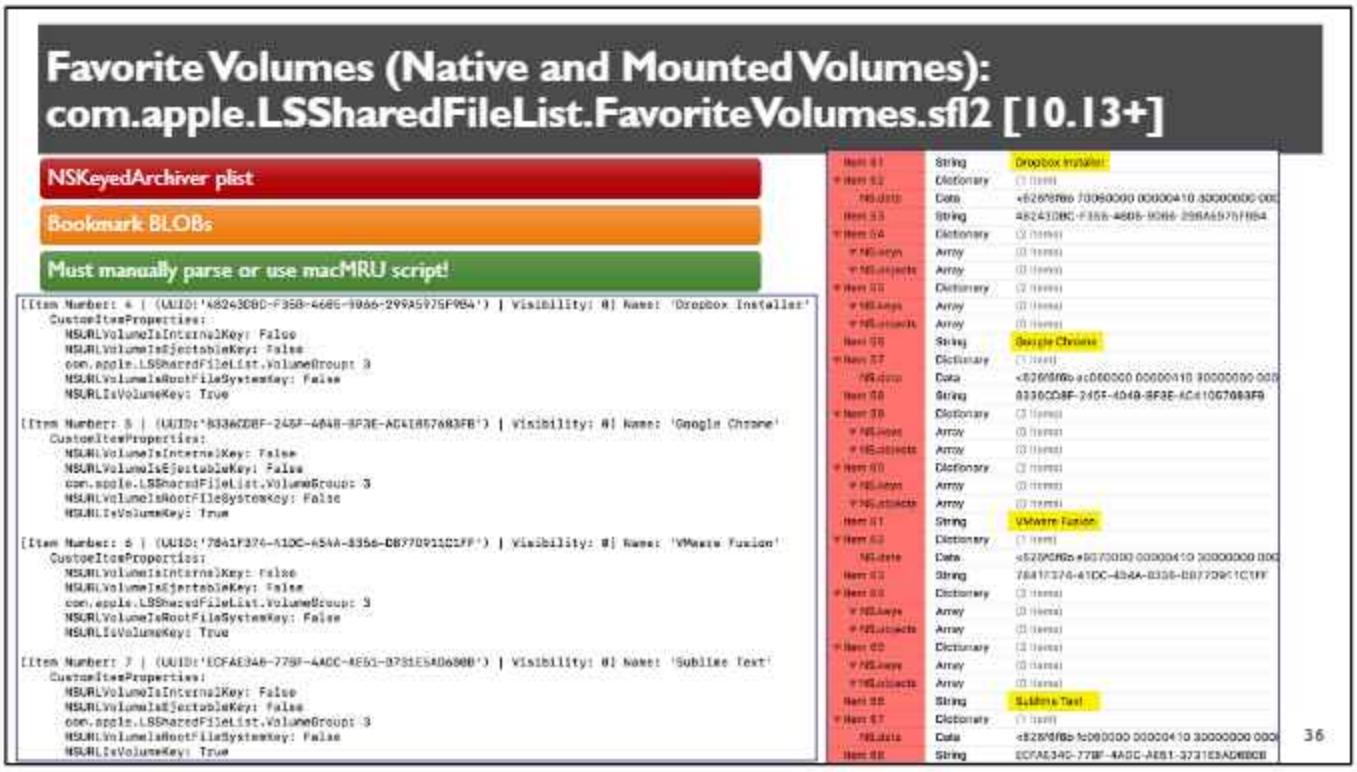
The `FXDesktopVolumePositions` key mainly stores the X and Y coordinates of volumes when mounted on the Desktop. This key can be useful for forensic analysts because it stores all the volumes that were mounted by the user. It unfortunately does not store the date when it was last mounted, but we may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

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

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

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

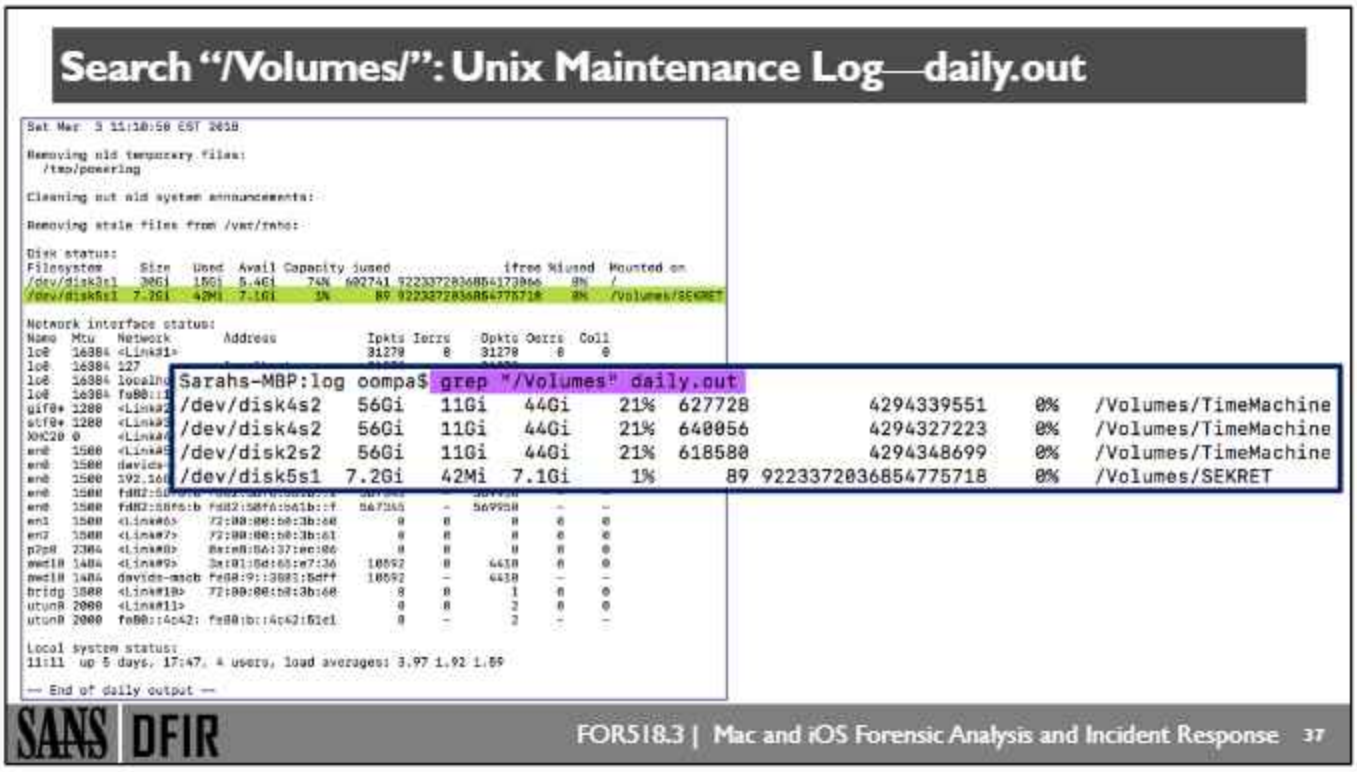
FXDesktopVolumePositions	Dictionary	(5 items)
Hopper Disassembler_0x1.2de30f9p+...	Dictionary	(4 items)
xRelative	Number	-79
ScreenID	Number	0
AnchorRelativeTo	Number	0
yRelative	Number	-75
VMware Fusion_0x1.2b208f48p+29	Dictionary	(4 items)
xRelative	Number	-79
ScreenID	Number	0
AnchorRelativeTo	Number	0
yRelative	Number	61
Macintosh HD_0x1.1de448p+29	Dictionary	(4 items)
xRelative	Number	-79
ScreenID	Number	0
AnchorRelativeTo	Number	0
yRelative	Number	-143
Impactor_0x1.1532cb38p+29	Dictionary	(4 items)
xRelative	Number	-79
ScreenID	Number	0
AnchorRelativeTo	Number	0
yRelative	Number	-7
Dropbox Installer_0x0p+0	Dictionary	(4 items)
xRelative	Number	277
ScreenID	Number	0
AnchorRelativeTo	Number	4
yRelative	Number	401



The sidebar volume information is found in the `com.apple.LSSharedFileList.FavoriteVolumes.sfl2` plist file in the `~/Library/Application Support/com.apple.sharedfilelist/` directory.

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

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

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

[Item Number: 6 | (UUID:'7841F374-41DC-454A-8356-0877D911C1FF') | Visibility: 0] Name: 'VMware Fusion'
    CustomItemProperties:
      NSURLVolumeIsInternalKey: False
      NSURLVolumeIsEjectableKey: False
      com.apple.LSSharedFileList.VolumeGroup: 3
      NSURLVolumeIsRootFileSystemKey: False
      NSURLIsVolumeKey: True

[Item Number: 7 | (UUID:'ECFAE348-778F-4ADC-AE51-3731E5AD680B') | Visibility: 0] Name: 'Sublime Text'
    CustomItemProperties:
      NSURLVolumeIsInternalKey: False
      NSURLVolumeIsEjectableKey: False
      com.apple.LSSharedFileList.VolumeGroup: 3
      NSURLVolumeIsRootFileSystemKey: False
      NSURLIsVolumeKey: True
```



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

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

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

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

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

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

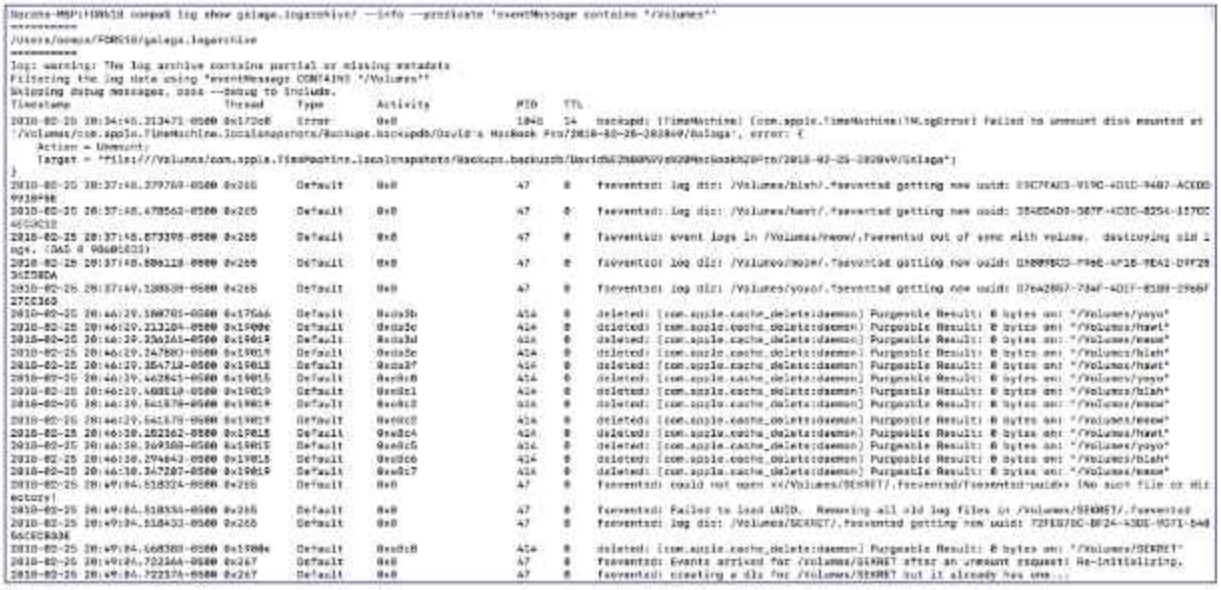
- **`daily.out`**: Contains output from temporary file removal, disk, network, and system status.
- **`weekly.out`**: Contains output from rebuilding the `whatis` database. This database contains a description of system commands.
- **`monthly.out`**: Contains output from rotating the fax logs and login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

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

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

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

Removing old temporary files:
  /tmp/powerlog

Cleaning out old system announcements:

Removing stale files from /var/rwho:

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

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

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

-- End of daily output --
```

Search "/Volumes/": system.log and Unified Logs

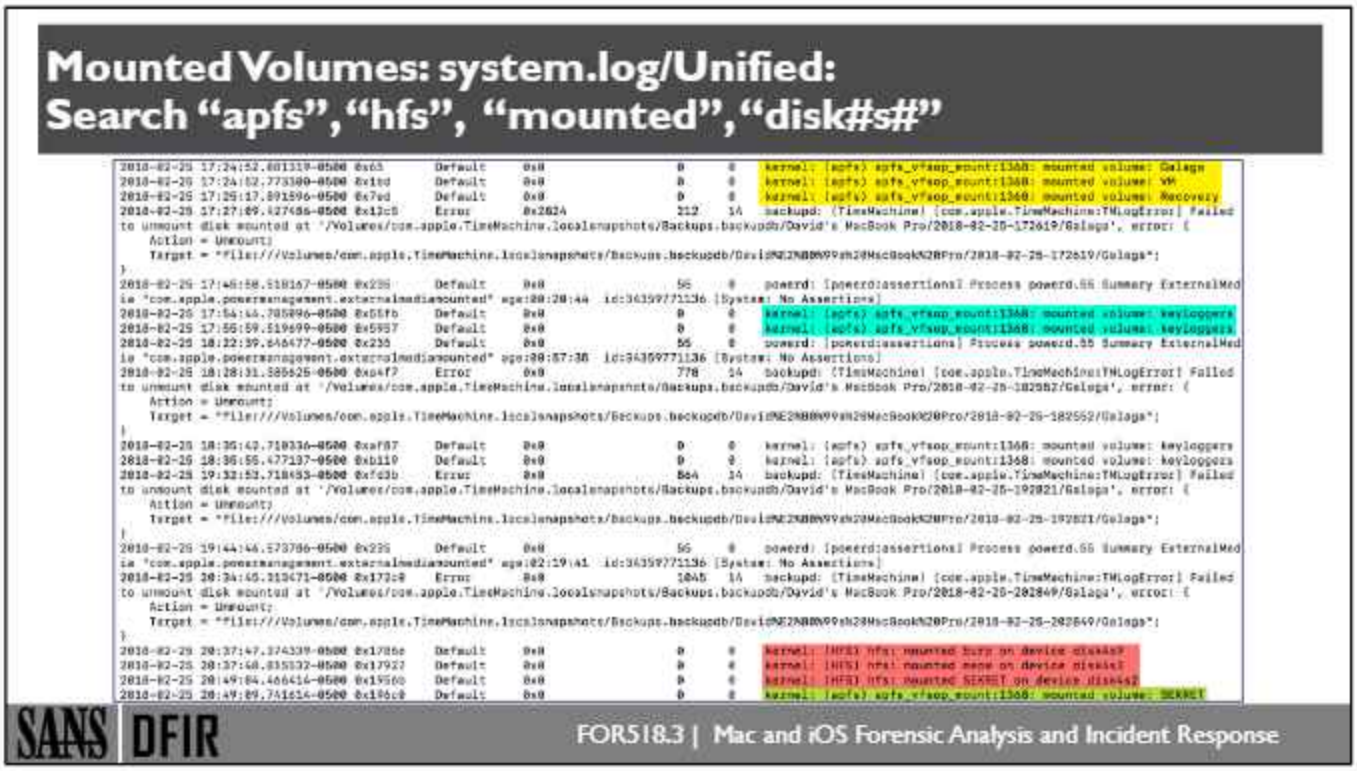


The term "`/Volumes/`" can be searched for in the `system.log` and `unified logs`.

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

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

```
Sarahs-MBP:FOR518 oompa$ log show galaga.logarchive/ --info --predicate 'eventMessage contains "/Volumes"'
==========
/Users/oompa/FOR518/galaga.logarchive
==========
log: warning: The log archive contains partial or missing metadata
Filtering the log data using "eventMessage CONTAINS "/Volumes""
Skipping debug messages, pass --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2018-02-25 20:34:45.313471-0500 0x172c0    Error       0x0                  1845   14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed to unmount disk mounted at
'/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-202849/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-202849/Galaga";
}
2018-02-25 20:37:48.379769-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/blah/.fseventsd getting new uuid: E9C7FAEB-919C-4D1C-9487-ACEDD
9910F5E
2018-02-25 20:37:48.478562-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/hawt/.fseventsd getting new uuid: 3B4ED4D9-307F-4C8C-8254-1E7CC
4EC3C12
2018-02-25 20:37:48.873398-0500 0x265      Default     0x0                  47     0    fseventsd: event logs in /Volumes/meow/.fseventsd out of sync with volume.  destroying old 1
ogs. (365 0 98601833)
2018-02-25 20:37:48.886110-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/meow/.fseventsd getting new uuid: DA8B9BCD-F96E-4F18-9E42-D9F25
34E50DA
2018-02-25 20:37:49.138530-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/yoyo/.fseventsd getting new uuid: D7642857-734F-4DEF-8180-2965F
27CE363
2018-02-25 20:46:29.188781-0500 0x17566    Default     0xda3b               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/yoyo"
2018-02-25 20:46:29.213184-0500 0x1900e    Default     0xda3c               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/hawt"
2018-02-25 20:46:29.236261-0500 0x19019    Default     0xda3d               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/meow"
2018-02-25 20:46:29.247803-0500 0x19019    Default     0xda3e               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/blah"
2018-02-25 20:46:29.354710-0500 0x19015    Default     0xda3f               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/hawt"
2018-02-25 20:46:29.462841-0500 0x19015    Default     0xe8c0               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/yoyo"
2018-02-25 20:46:29.488510-0500 0x19019    Default     0xe8c1               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/blah"
2018-02-25 20:46:29.541578-0500 0x19019    Default     0xe8c2               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/meow"
2018-02-25 20:46:29.541578-0500 0x19019    Default     0xe8c2               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/meow"
2018-02-25 20:46:30.152362-0500 0x19015    Default     0xe8c4               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/hawt"
2018-02-25 20:46:30.269380-0500 0x19015    Default     0xe8c5               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/yoyo"
2018-02-25 20:46:30.294643-0500 0x19015    Default     0xe8c6               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/blah"
2018-02-25 20:46:30.347207-0500 0x19019    Default     0xe8c7               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/meow"
2018-02-25 20:49:04.518324-0500 0x265      Default     0x0                  47     0    fseventsd: could not open <</Volumes/SEKRET/.fseventsd/fseventsd-uuid>> (No such file or dir
ectory)
2018-02-25 20:49:04.518334-0500 0x265      Default     0x0                  47     0    fseventsd: Failed to load UUID.  Removing all old log files in /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:04.518433-0500 0x265      Default     0x0                  47     0    fseventsd: log dir: /Volumes/SEKRET/.fseventsd getting new uuid: 72FEB78C-BF24-43DE-9D71-540
56CECB33E
2018-02-25 20:49:04.668388-0500 0x1900e    Default     0xe8c8               414    0    deleted: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:04.722366-0500 0x267      Default     0x0                  47     0    fseventsd: Events arrived for /Volumes/SEKRET after an unmount request! Re-initializing.
2018-02-25 20:49:04.722376-0500 0x267      Default     0x0                  47     0    fseventsd: creating a dls for /Volumes/SEKRET but it already has one...
```



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

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

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

```
2018-02-25 17:24:52.881319-0500 0x65       Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: Galaga
2018-02-25 17:24:52.773300-0500 0x1bd      Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: VM
2018-02-25 17:25:17.891896-0500 0x7ed      Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: Recovery
2018-02-25 17:27:09.427486-0500 0x12c5     Error       0x2824               212    14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed
to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-172619/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-172619/Galaga";
}
2018-02-25 17:45:50.518167-0500 0x235      Default     0x0                  55     0    powerd: [powerd:assertions] Process powerd.55 Summary ExternalMed
ia "com.apple.powermanagement.externalmediamounted" age:00:20:44  id:34359771136 [System: No Assertions]
2018-02-25 17:54:44.785096-0500 0x55fb     Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: keyloggers
2018-02-25 17:55:59.519699-0500 0x5957     Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: keyloggers
2018-02-25 18:22:39.646477-0500 0x235      Default     0x0                  55     0    powerd: [powerd:assertions] Process powerd.55 Summary ExternalMed
ia "com.apple.powermanagement.externalmediamounted" age:00:57:35  id:34359771136 [System: No Assertions]
2018-02-25 18:28:31.385625-0500 0xa4f7     Error       0x0                  778    14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed
to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-182552/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-182552/Galaga";
}
2018-02-25 18:35:42.710336-0500 0xaf87     Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: keyloggers
2018-02-25 18:35:55.477137-0500 0xb119     Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: keyloggers
2018-02-25 19:32:53.718453-0500 0xfd3b     Error       0x0                  864    14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed
to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-192821/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-192821/Galaga";
}
2018-02-25 19:44:46.573786-0500 0x235      Default     0x0                  55     0    powerd: [powerd:assertions] Process powerd.55 Summary ExternalMed
ia "com.apple.powermanagement.externalmediamounted" age:02:19:41  id:34359771136 [System: No Assertions]
2018-02-25 20:34:45.313471-0500 0x172c0    Error       0x0                  1045   14   backupd: (TimeMachine) [com.apple.TimeMachine:TMLogError] Failed
to unmount disk mounted at '/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David's MacBook Pro/2018-02-25-202849/Galaga', error: {
    Action = Unmount;
    Target = "file:///Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/David%E2%80%99s%20MacBook%20Pro/2018-02-25-202849/Galaga";
}
2018-02-25 20:37:47.374339-0500 0x1766e    Default     0x0                  0      0    kernel: (HFS) hfs: mounted burp on device disk4s9
2018-02-25 20:37:48.815532-0500 0x17927    Default     0x0                  0      0    kernel: (HFS) hfs: mounted meow on device disk4s3
2018-02-25 20:49:04.466414-0500 0x1956b    Default     0x0                  0      0    kernel: (HFS) hfs: mounted SEKRET on device disk4s2
2018-02-25 20:49:09.741614-0500 0x196c0    Default     0x0                  0      0    kernel: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
```

Mounted Volumes: system.log/Unified: Search by Volume Name

```
Sarahs-MBP:FOR518 ocmpa$ log show galaga.logarchive/ --info --predicate 'eventMessage contains "SEKRET"' --style syslog
==========
/Users/ocmpa/FOR518/galaga.logarchive
==========
log: warning: The log archive contains partial or missing metadata
Filtering the log data using "eventMessage CONTAINS "SEKRET""
Skipping debug messages; pass --debug to include.
Timestamp                       (process)[PID]
2018-02-25 20:48:58.713365-0500  localhost diskmanagementd[125?]: diskmanagement: execve(2) pid=125? /System/Library/Filesystems/hfs.fs/Contents/Resources/newfs_hfs -J -v
 SEKRET /dev/rdisk4s2 .
2018-02-25 20:49:04.466416-0500  localhost kernel[0]: (HFS) hfs: mounted SEKRET on device disk4s2
2018-02-25 20:49:04.518324-0500  localhost fseventsd[47]: could not open <</Volumes/SEKRET/.fseventsd/fseventsd-uuid>> (No such file or directory)
2018-02-25 20:49:04.518334-0500  localhost fseventsd[47]: Failed to load UUID.  Removing all old log files in /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:04.518433-0500  localhost fseventsd[47]: log dir: /Volumes/SEKRET/.fseventsd getting new uuid: 72FE87BC-8F24-43DE-9D71-54B5ACEC8336
2018-02-25 20:49:04.668380-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:04.722366-0500  localhost fseventsd[47]: Events arrived for /Volumes/SEKRET after an unmount request! Re-initializing.
2018-02-25 20:49:04.722376-0500  localhost fseventsd[47]: creating a dls for /Volumes/SEKRET but it already has one...
2018-02-25 20:49:06.473870-0500  localhost kernel[0]: (HFS) hfs: unmount initiated on SEKRET on device disk4s2
2018-02-25 20:49:06.682182-0500  localhost fseventsd[47]: disk logger: failed to open output file /Volumes/SEKRET/.fseventsd/0000000027b86e (No such file or directory).
 mount point /Volumes/SEKRET/.fseventsd .
2018-02-25 20:49:06.682361-0500  localhost fseventsd[47]: disk logger: failed to open output file /Volumes/SEKRET/.fseventsd/0000000027b86e (No such file or directory).
 mount point /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:08.719848-0500  localhost diskmanagementd[1276]: diskmanagement: execve(2) pid=1276 /System/Library/Filesystems/apfs.fs/Contents/Resources/newfs_apfs -A
-i -E -S frogger13 -v SEKRET disk4 .
2018-02-25 20:49:09.741614-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-25 20:49:09.794588-0500  localhost fseventsd[47]: could not open <</Volumes/SEKRET/.fseventsd/fseventsd-uuid>> (No such file or directory)
2018-02-25 20:49:09.794696-0500  localhost fseventsd[47]: Failed to load UUID.  Removing all old log files in /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:09.794711-0500  localhost fseventsd[47]: log dir: /Volumes/SEKRET/.fseventsd getting new uuid: AD4CCAB3-6EDE-4FC8-9B21-D98A18C3CAA9
2018-02-25 20:49:09.798444-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:09.816415-0500  localhost storagekitd[1228]: Erase Complete. Mount Point: /Volumes/SEKRET
2018-02-25 20:49:09.830168-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:09.858039-0500  localhost storagekitd[1228]: Recache Complete, Mount Point: /Volumes/SEKRET
2018-02-25 20:49:58.326722-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-27 18:20:25.719892-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-27 18:20:28.828216-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-27 18:22:13.848939-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-28 18:58:39.662188-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-28 18:58:39.784029-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
```


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

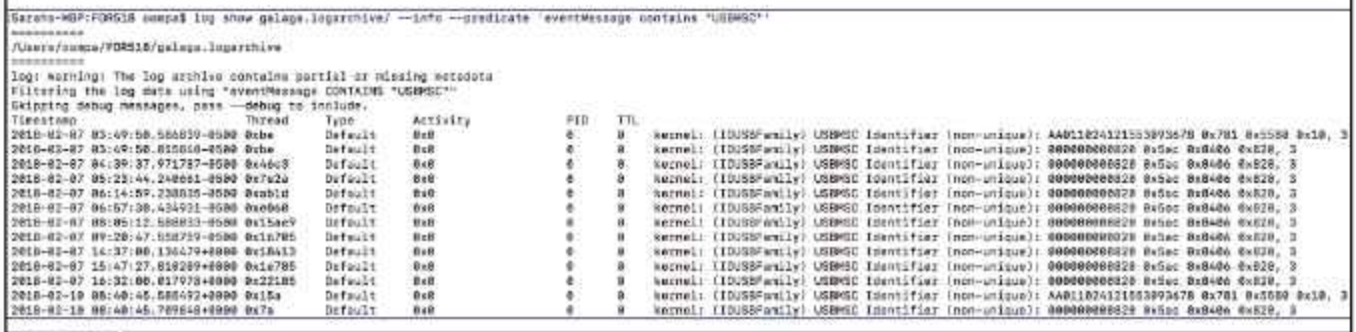
The screenshot above shows the activity for a volume named "SEKRET". Some of the items you may find in addition to mounting activity and volume use is volume formatting—note the highlighted entries. These are the artifacts created of an encrypted APFS (from an HFS+ volume) volume named "SEKRET". On 10.13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

```
Sarahs-MBP:FOR518 oompa$ log show galaga.logarchive/ --info --predicate 'eventMessage contains "SEKRET"' --style syslog
==========
/Users/oompa/FOR518/galaga.logarchive
==========
log: warning: The log archive contains partial or missing metadata
Filtering the log data using "eventMessage CONTAINS "SEKRET""
Skipping debug messages, pass --debug to include.
Timestamp                       (process)[PID]
2018-02-25 20:48:58.712365-0500  localhost diskmanagementd[1257]: diskmanagement: execve(2) pid=1267 /System/Library/Filesystems/hfs.fs/Contents/Resources/newfs_hfs -J -v
 SEKRET /dev/rdisk4s2 .
2018-02-25 20:49:04.466414-0500  localhost kernel[0]: (HFS) hfs: mounted SEKRET on device disk4s2
2018-02-25 20:49:04.518324-0500  localhost fseventsd[47]: could not open <</Volumes/SEKRET/.fseventsd/fseventsd-uuid>> (No such file or directory)
2018-02-25 20:49:04.518334-0500  localhost fseventsd[47]: Failed to load UUID.  Removing all old log files in /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:04.518433-0500  localhost fseventsd[47]: log dir: /Volumes/SEKRET/.fseventsd getting new uuid: 72FEB7BC-8F24-43DE-9D71-54856CEC833E
2018-02-25 20:49:04.668388-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:04.722366-0500  localhost fseventsd[47]: Events arrived for /Volumes/SEKRET after an unmount request! Re-initializing.
2018-02-25 20:49:04.722376-0500  localhost fseventsd[47]: creating a dls for /Volumes/SEKRET but it already has one...
2018-02-25 20:49:06.473078-0500  localhost kernel[0]: (HFS) hfs: unmount initiated on SEKRET on device disk4s2
2018-02-25 20:49:06.602182-0500  localhost fseventsd[47]: disk logger: failed to open output file /Volumes/SEKRET/.fseventsd/000000000027b06e (No such file or directory).
 mount point /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:06.602361-0500  localhost fseventsd[47]: disk logger: failed to open output file /Volumes/SEKRET/.fseventsd/000000000027b06e (No such file or directory).
 mount point /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:08.719840-0500  localhost diskmanagementd[1276]: diskmanagement: execve(2) pid=1276 /System/Library/Filesystems/apfs.fs/Contents/Resources/newfs_apfs -A
-i -E -S frogger13 -v SEKRET disk5 .
2018-02-25 20:49:09.741614-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-25 20:49:09.794588-0500  localhost fseventsd[47]: could not open <</Volumes/SEKRET/.fseventsd/fseventsd-uuid>> (No such file or directory)
2018-02-25 20:49:09.794596-0500  localhost fseventsd[47]: Failed to load UUID.  Removing all old log files in /Volumes/SEKRET/.fseventsd
2018-02-25 20:49:09.794711-0500  localhost fseventsd[47]: log dir: /Volumes/SEKRET/.fseventsd getting new uuid: AD6ECA53-6EDE-4FC8-9821-D9BA10C3C6A9
2018-02-25 20:49:09.798644-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:09.816415-0500  localhost storagekitd[1220]: Erase Complete, Mount Point: /Volumes/SEKRET
2018-02-25 20:49:09.830160-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-25 20:49:09.858039-0500  localhost storagekitd[1220]: Recache Complete, Mount Point: /Volumes/SEKRET
2018-02-25 20:49:58.326722-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-27 18:20:25.719892-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-27 18:20:25.828215-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-27 18:22:13.848939-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
2018-02-28 18:58:39.652188-0500  localhost kernel[0]: (apfs) apfs_vfsop_mount:1368: mounted volume: SEKRET
2018-02-28 18:58:39.784029-0500  localhost deleted[414]: [com.apple.cache_delete:daemon] Purgeable Result: 0 bytes on: "/Volumes/SEKRET"
```

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

Serial Number, Vendor ID, Product ID, Version

- 10.12+: Unified Logs
- **USBMSC Identifier (non-unique): FBF1011220504638 0x90c 0x1000 0x1100**



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

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

```
USBMSC Identifier (non-unique): FBF1011220504638 0x90c 0x1000 0x1100
```

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

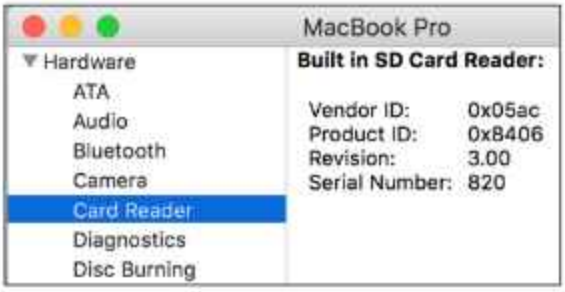
- Vendor ID: 0x090c (Silicon Motion, Inc.)
- Product ID: 0x1000
- Version: 11.00

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

```
Sarahs-MBP:FOR518 oompa$ log show gaiaga.logarchive/ --info --predicate 'eventMessage contains "USBMSC"'
==========
/Users/oompa/FOR518/gaiaga.logarchive
==========
Log: warning: The log archive contains partial or missing metadata
Filtering the log data using "eventMessage CONTAINS "USBMSC""
Skipping debug messages, pass --debug to include.
Timestamp                       Thread     Type        Activity            PID    TTL
2018-02-07 03:49:50.586839-0500 0xbe       Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): AA01024121553093678 0x781 0x5500 0x10, 3
2018-02-07 03:49:50.615048-0500 0xbe       Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 04:39:37.971787-0500 0x46c3     Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 05:23:44.240661-0500 0x7d2a     Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 06:14:59.238935-0500 0xab1d     Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 06:57:30.434931-0500 0xe060     Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 08:05:12.588033-0500 0x15ae9    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 09:20:47.558750-0500 0x16705    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 14:37:00.136479+0000 0x18413    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 15:47:27.810269+0000 0x1e785    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-07 16:32:00.617978+0000 0x22185    Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
2018-02-10 08:40:45.595492+0000 0x15a      Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): AA01024121553093678 0x781 0x5500 0x10, 3
2018-02-10 08:40:45.789848+0000 0x7a       Default     0x0                 0      0    kernel: (IOUSBFamily) USBMSC Identifier (non-unique): 0000000000820 0x5ac 0x8406 0x820, 3
```

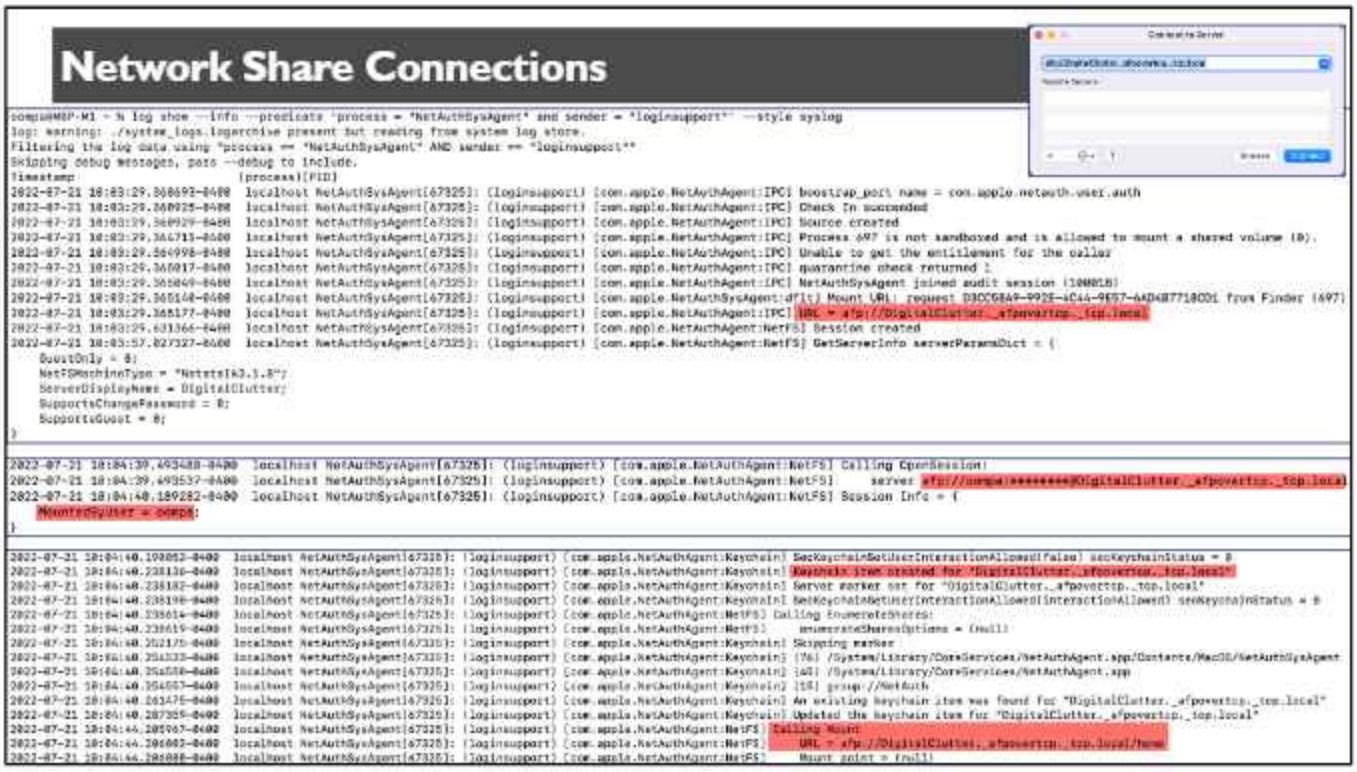
USBMSC Caveat: SD Card Reader

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



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

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



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

It shows the NAS server (afp://DigitalClutter), user (oompa) logging in and share folder (/home) being mounted.

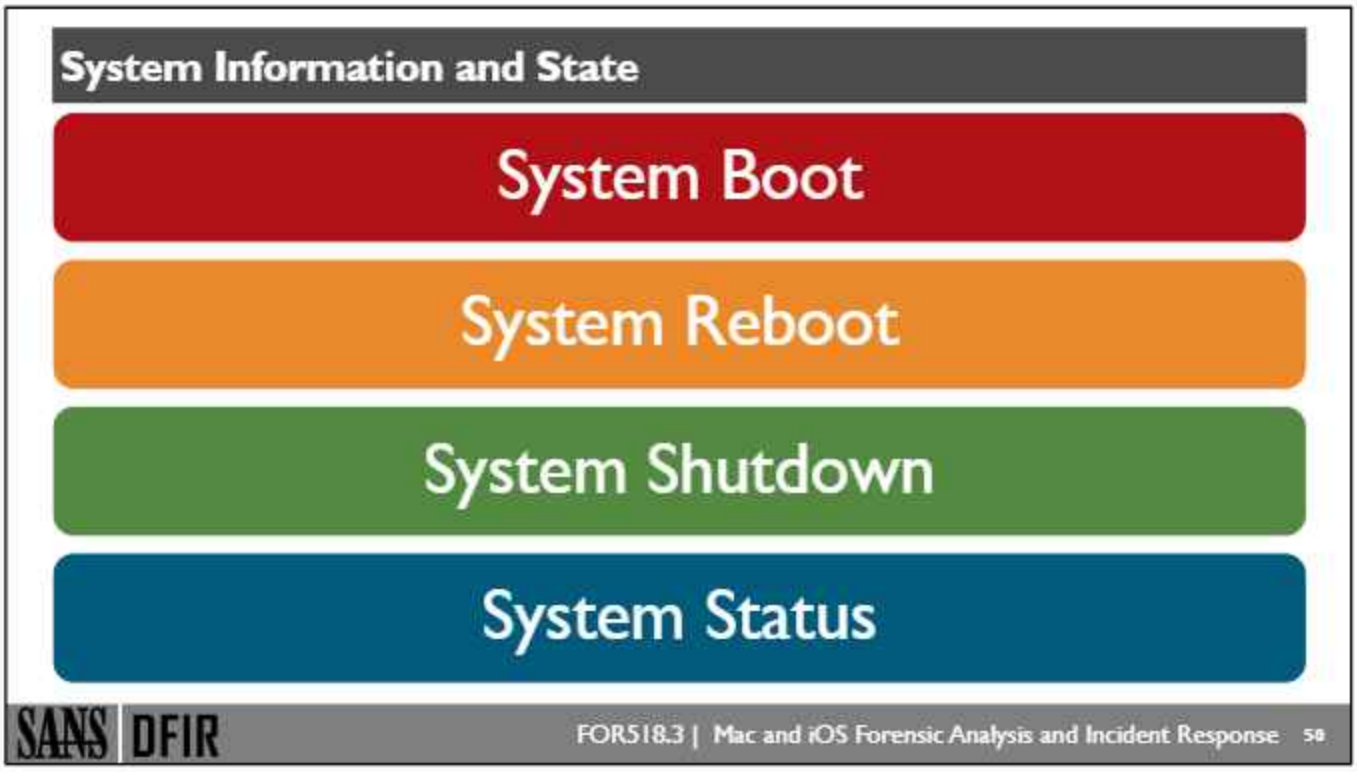
This example was created using the "Connect to Server" functionality of Finder.

Older logs may use "afpfs", "smbfs", "smb://" to identify network shares.

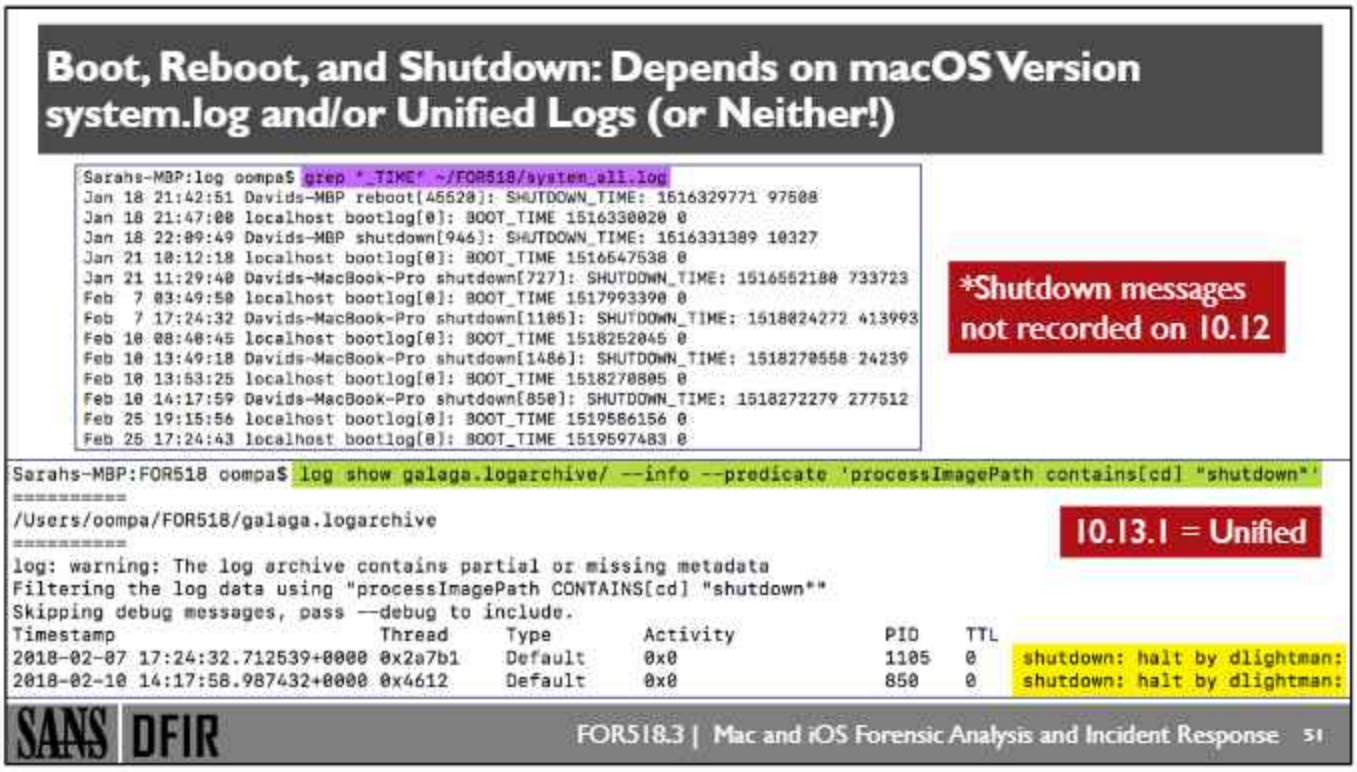
```
2022-07-21 10:04:40.190852-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] SecKeychainSetUserInteractionAllowed(false) secKeychainStatus = 0
2022-07-21 10:04:40.238136-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] Keychain item created for "DigitalClutter._afpovertcp._tcp.local"
2022-07-21 10:04:40.238182-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] Server marker set for "DigitalClutter._afpovertcp._tcp.local"
2022-07-21 10:04:40.238190-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] SecKeychainSetUserInteractionAllowed(interactionAllowed) secKeychainStatus = 0
2022-07-21 10:04:40.238614-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Calling EnumerateShares:
2022-07-21 10:04:40.238619-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS]     enumerateSharesOptions = (null)
2022-07-21 10:04:40.252176-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] Skipping marker
2022-07-21 10:04:40.254533-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] (76) /System/Library/CoreServices/NetAuthAgent.app/Contents/MacOS/NetAuthSysAgent
2022-07-21 10:04:40.254550-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] (45) /System/Library/CoreServices/NetAuthAgent.app
2022-07-21 10:04:40.254557-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] (15) group://NetAuth
2022-07-21 10:04:40.261475-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] An existing keychain item was found for "DigitalClutter._afpovertcp._tcp.local"
2022-07-21 10:04:40.287359-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:Keychain] Updated the keychain item for "DigitalClutter._afpovertcp._tcp.local"
2022-07-21 10:04:44.285967-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Calling Mount
2022-07-21 10:04:44.286082-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS]     URL = afp://DigitalClutter._afpovertcp._tcp.local/home
2022-07-21 10:04:44.286088-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS]     Mount point = (null)
```

```
2022-07-21 10:04:39.493488-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Calling OpenSession:
2022-07-21 10:04:39.493537-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS]     server afp://oompa:********@DigitalClutter._afpovertcp._tcp.loca
2022-07-21 10:04:40.189282-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Session Info = {
    MountedByUser = oompa;
}
```

```
oompa@MBP-M1 ~ % log show --info --predicate 'process = "NetAuthSysAgent" and sender = "loginsupport"' --style syslog
log: warning: ./system_logs.logarchive present but reading from system log store.
Filtering the log data using "process == "NetAuthSysAgent" AND sender == "loginsupport""
Skipping debug messages, pass --debug to include.
Timestamp                       (process)[PID]
2022-07-21 10:03:29.368693-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:IPC] bootstrap_port name = com.apple.netauth.user.auth
2022-07-21 10:03:29.368926-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:IPC] Check In succeeded
2022-07-21 10:03:29.368929-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:IPC] Source created
2022-07-21 10:03:29.364711-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:IPC] Process 697 is not sandboxed and is allowed to mount a shared volume (0).
2022-07-21 10:03:29.364998-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:IPC] Unable to get the entitlement for the caller
2022-07-21 10:03:29.365017-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:IPC] quarantine check returned 1
2022-07-21 10:03:29.365049-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:IPC] NetAuthSysAgent joined audit session (100018)
2022-07-21 10:03:29.365140-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthSysAgent:dflt] Mount URL: request D3CC58A9-992E-4C44-9E57-6AD4B7718CD1 from Finder (697)
2022-07-21 10:03:29.365177-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:IPC] URL = afp://DigitalClutter._afpovertcp._tcp.local
2022-07-21 10:03:29.631366-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS] Session created
2022-07-21 10:03:57.027327-0400  localhost NetAuthSysAgent[67325]: (loginsupport) [com.apple.NetAuthAgent:NetFS] GetServerInfo serverParamsDict = {
    GuestOnly = 0;
    NetFSMachineType = "Netatalk3.1.6";
    ServerDisplayName = DigitalClutter;
    SupportsChangePassword = 0;
    SupportsGuest = 0;
}
```



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



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

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

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

Older examples are below from the system.log.

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

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

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

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

macOS Shutdowns and Reboots

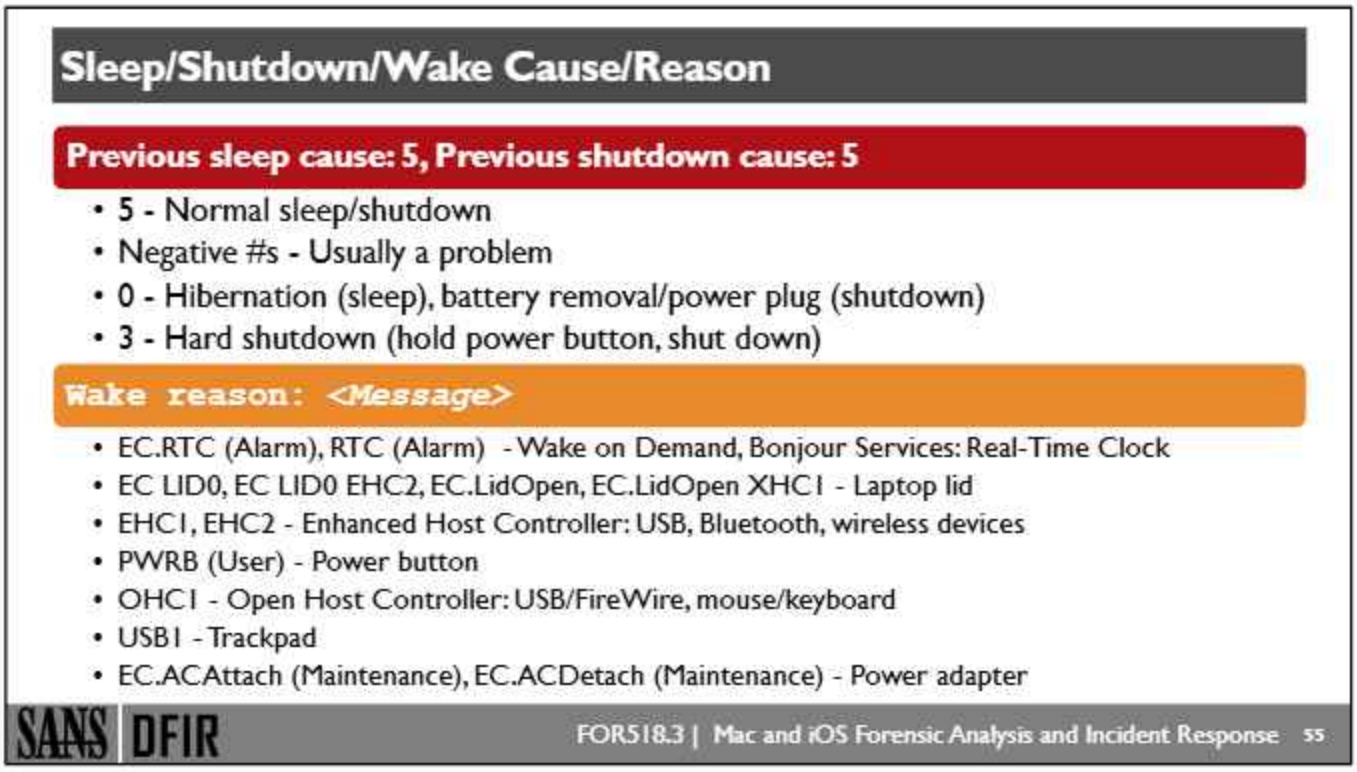
```
log show --predicate 'eventMessage
contains "com.apple.system.loginwindow"
and eventMessage contains
"SessionAgentNotificationCenter"'
```

SANS DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 53

Messages associated with '`SessionAgentNotificationCenter`' are easy to interpret to determine if a user shutdown, restart, or cancelled a shutdown/restart/logoff of a system.

- `com.apple.system.loginwindow.shutdownInitiated` – User chose to shutdown system
- `com.apple.system.loginwindow.logoutcancelled` – User canceled the shutdown (or restart or logoff)
- `com.apple.system.loginwindow.restartinitiated` – User chose to restart system

Reference:
https://www.mac4n6.com/blog/2020/4/26/analysis-of-apple-unified-logs-quarantine-edition-entry-4-its-login-week

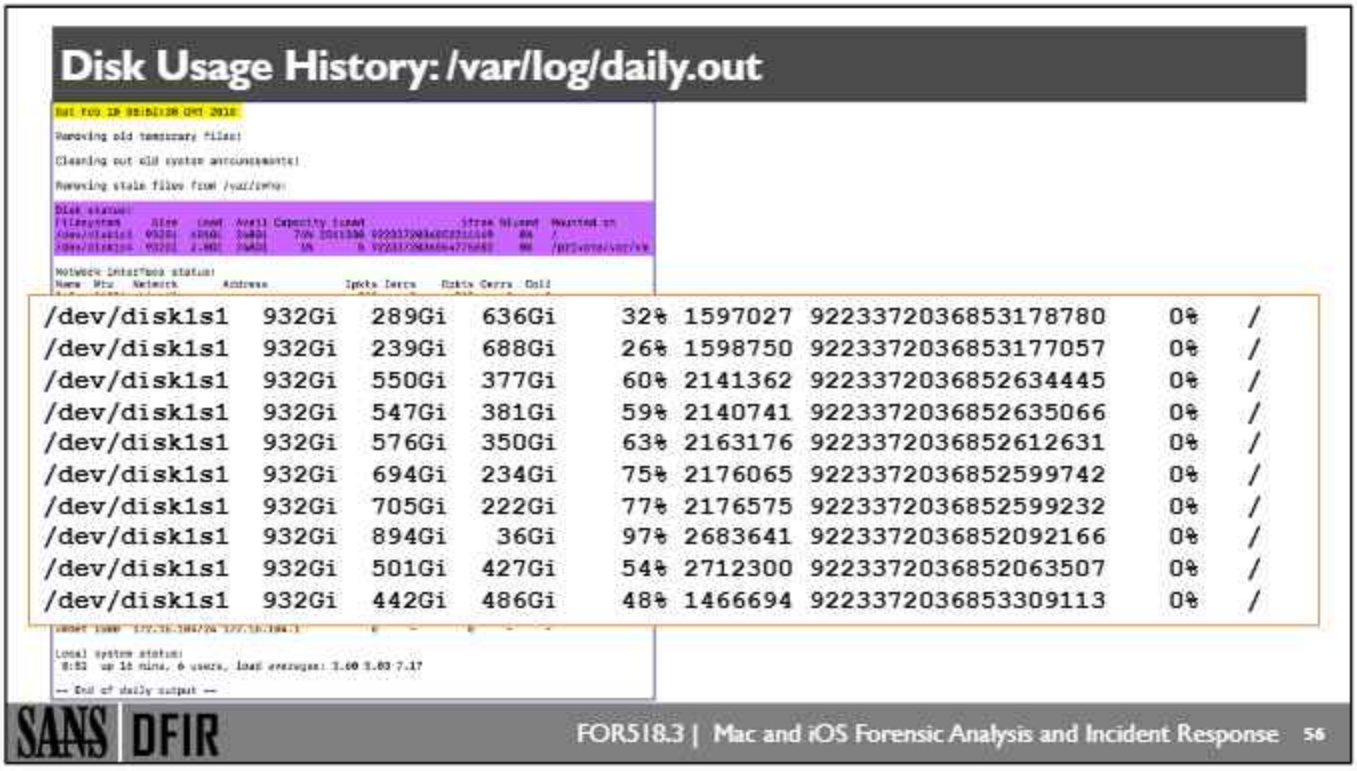


The system records reasons the computer has been put to sleep, which is known as "`Sleep Cause`" or "`Shutdown Cause`".

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

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

Disk Usage History: /var/log/daily.out



```
/dev/disk1s1   932Gi   289Gi   636Gi     32% 1597027 9223372036853178780     0%    /
/dev/disk1s1   932Gi   239Gi   688Gi     26% 1598750 9223372036853177057     0%    /
/dev/disk1s1   932Gi   550Gi   377Gi     60% 2141362 9223372036852634445     0%    /
/dev/disk1s1   932Gi   547Gi   381Gi     59% 2140741 9223372036852635066     0%    /
/dev/disk1s1   932Gi   576Gi   350Gi     63% 2163176 9223372036852612631     0%    /
/dev/disk1s1   932Gi   694Gi   234Gi     75% 2176065 9223372036852599742     0%    /
/dev/disk1s1   932Gi   705Gi   222Gi     77% 2176575 9223372036852599232     0%    /
/dev/disk1s1   932Gi   894Gi    36Gi     97% 2683641 9223372036852092166     0%    /
/dev/disk1s1   932Gi   501Gi   427Gi     54% 2712300 9223372036852063507     0%    /
/dev/disk1s1   932Gi   442Gi   486Gi     48% 1466694 9223372036853309113     0%    /
```


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

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

Removing old temporary files:

Cleaning out old system announcements:

Removing stale files from /var/rwho:

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

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

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

-- End of daily output --
```

Application Bundles: Package Contents

SANS DFIR | FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 58

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

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

- Executable Code
- Resource Files
- Support Files

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

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

References:
Bundle Programming Guide: Apple Developer Documentation
https://developer.apple.com/library/archive/documentation/CoreFoundation/Conceptual/CFBundles/AboutBundles/AboutBundles.html

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

macOS Application Bundles: Information Property List—Info.plist

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

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

The example shows an `Info.plist` file for the VMware Fusion application, version 11.5.3. Other items in the file may be optional but still useful to an investigator to determine the purpose of a specific application. The "`Application Category`" is how the developer determines what type of application it is, while the "`Document types`" and "`URL types`" determine what files or URLs this program supports. The application may also state what versions of macOS it will run on, listing minimum and system versions and architecture types.

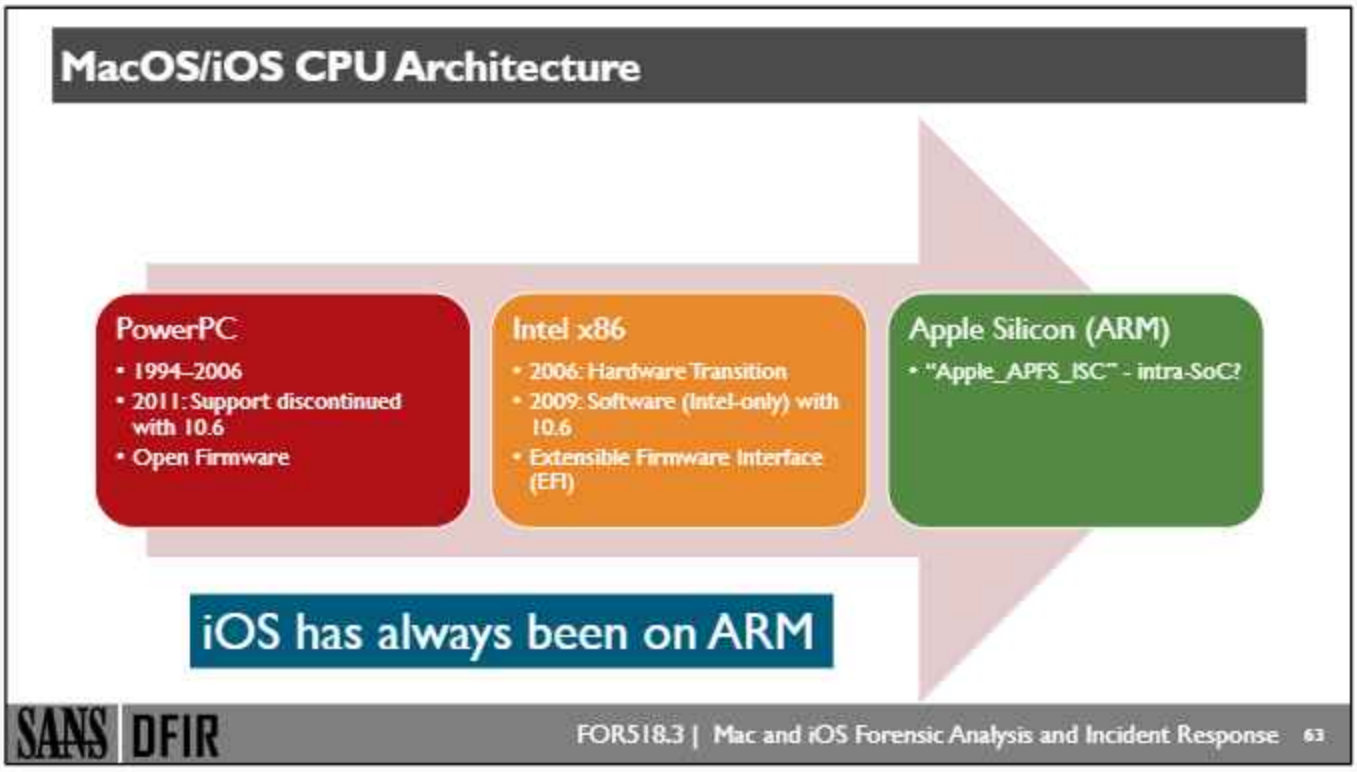
Worth noting are the "`Privacy`" keys—this may help in determining the use of an unknown application.

iOS Application Bundles (Physical)
Information Property List—iTunesMetadata.plist

Physical Location: /private/var/containers/Bundle/Application/<GUID>/iTunesMetadata.plist

Similar information for iOS apps is stored in their iTunesMetadata.plist files including the purchase date and user information.

▼ Root	Dictionary	(24 items)
DeviceBasedVPP	Boolean	NO
artistName	String	Philz Coffee
bundleVersion	String	1562797260
▼ com.apple.iTunesStore.downloadInfo	Dictionary	(2 items)
▼ accountInfo	Dictionary	(6 items)
AltDSID	String	001792[redacted]
AppleID	String	oompa@csh.rit.edu
DSPersonID	Number	24,7[redacted]
DownloaderID	Number	0
FamilyID	Number	0
PurchaserID	Number	24,7[redacted]
purchaseDate	String	2019-02-18T14:25:37Z
gameCenterEnabled	Boolean	NO
gameCenterEverEnabled	Boolean	NO
genre	String	Food & Drink
genreId	Number	6,023
hasMessagesExtension	Boolean	NO
is-auto-download	Boolean	NO
is-purchased-redownload	Boolean	NO
itemId	Number	1,298,298,285
itemName	String	Philz Coffee
kind	String	software
launchProhibited	Boolean	NO
▼ rating	Dictionary	(2 items)
label	String	4+
rank	Number	100
s	Number	143,441
sideLoadedDeviceBasedVPP	Boolean	NO
softwareVersionBundleId	String	com.philzcoffee.app
softwareVersionExternalIdentifier	Number	831,982,173
sourceApp	String	com.apple.AppStore
storeCohort	String	7|date=1550498400000&sf=14
▶ subgenre	Array	(0 items)
variantID	String	1:iPhone10,6:11



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

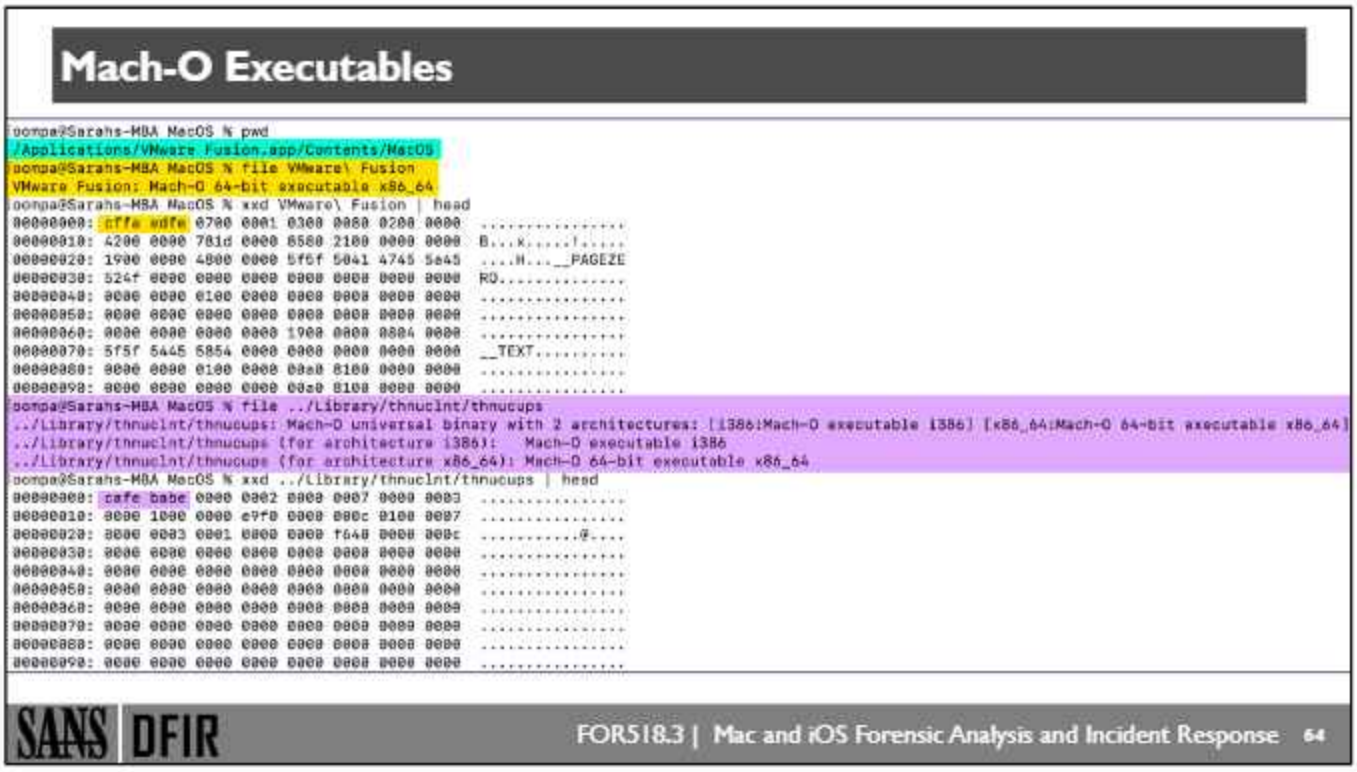
PowerPC (Performance Optimization With Enhanced RISC – Performance Computing) was used from 1994 to 2006. PowerPC was no longer supported with OS X Snow Leopard (10.6). This architecture used the Open Firmware interface, similar to a PC's BIOS. The last version of OS X that was PowerPC-only was OS X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

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

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

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

With Apple Silicon introduced at WWDC 2020, the ARM M1 was introduced later in 2020.



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

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

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

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

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

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

References:

macOS Mach-O File Format Reference: Apple Developer Documentation
https://developer.apple.com/library/archive/documentation/DeveloperTools/Conceptual/MachOTopics/0-Introduction/introduction.html
https://appleinsider.com/articles/08/10/28/road_to_mac_os_x_snow_leopard_64_bit_to_the_kernel.html

Extensions (Kernel and System)

Low-Level Device Drivers, Network Filters, File Systems…Keyloggers? … Malware?

Kernel Extensions (Kext) - Dynamically loaded kernel code

- `kmutil showloaded` (`kextstat` is deprecated)
- 10.13 – Introduced User-Approved (Kernel) Extension Loading
- /private/var/db/loadedkextmt.plist

System Extensions – Introduced in 10.15

- `systemextensionsctl list`
- DriverKit, Network, Endpoint Security
- Works via specific entitlements, Apple controls acquiring of these – slow adoption rates

Directories (Pay additional attention to those with **)

- /Library/Apple/System/Library/Extensions
- /System/Library/Extensions/
- /Library/Extensions/
- **/Library/StagedExtensions/
- **/Library/SystemExtensions/
- **/Library/<*Filesystems/macfuse.fs/Contents*>/Extensions

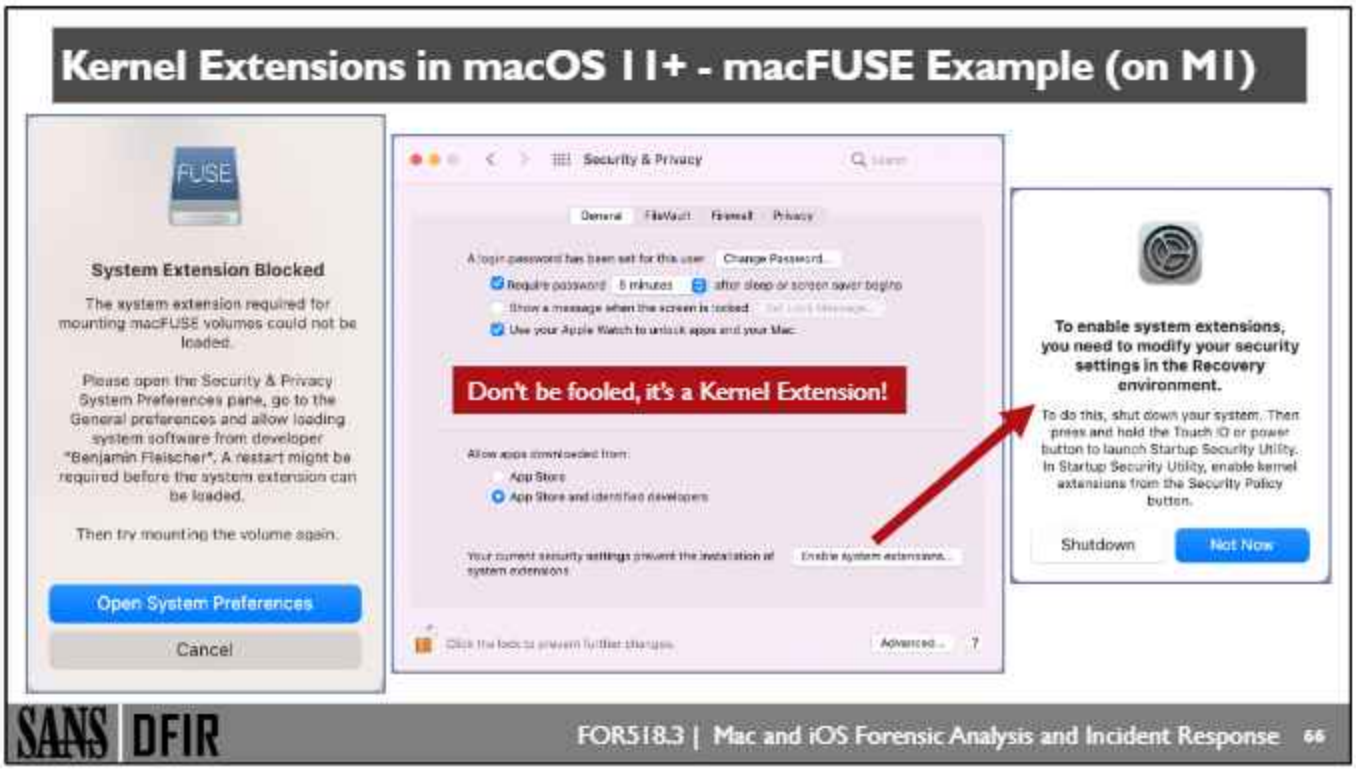
SANS DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 65

Kernel extensions are similar to loadable kernel modules on Linux systems. They are often used as device drivers, network filters, or support for various file systems. They can also be used maliciously, thus the reason for System Extensions were introduced.

You can use the `kextstat`/`kmutil`/`systemextensionsctl` commands on a live system to view the status of the kernel/system extensions. To see what third-party kernel extensions were loaded on a dead system, look for `/private/var/db/loadedkextmt.plist`.

References:
https://developer.apple.com/system-extensions/
https://developer.apple.com/support/kernel-extensions/

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



Software that still uses kernel extensions, like macFUSE in this example, can still run but will need to be allowed to do so. On an M1 Mac this requires an admin user to restart the system, go into Recover mode, and downgrade the security to allow use of kernel extensions using the Startup Security Utility.

Reference:
https://support.apple.com/guide/deployment-reference-macos/kernel-extensions-in-macos-apd37565d329/web

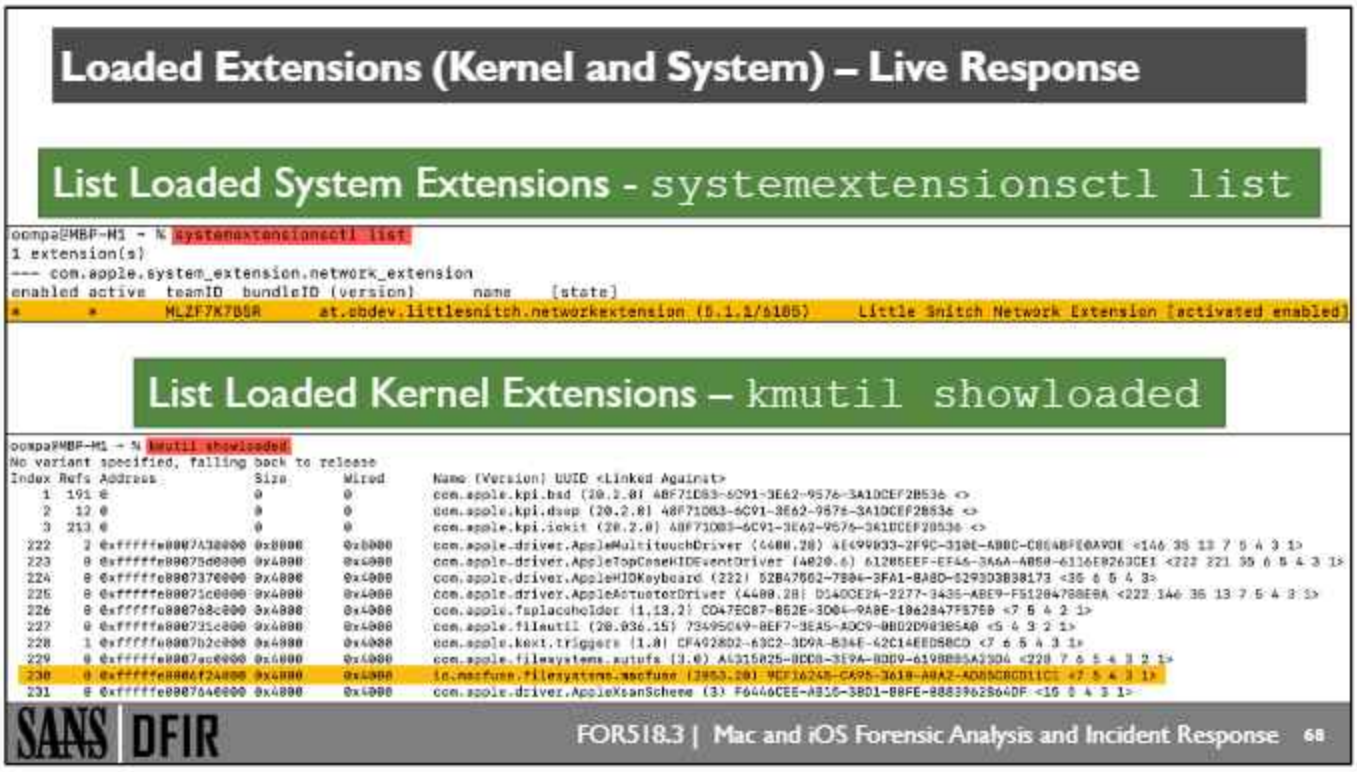
Extensions (Kernel and System): */Extensions/* Directories

```
oompa@MBP-M1 26382148-1D0B-4B5A-BD3C-B24E2EE550C2 % pwd
/Library/SystemExtensions/26382148-1D0B-4B5A-BD3C-B24E2EE550C2
oompa@MBP-M1 26382148-1D0B-4B5A-BD3C-B24E2EE550C2 % find .
.
./at.obdev.littlesnitch.networkextension.systemextension
./at.obdev.littlesnitch.networkextension.systemextension/Contents
./at.obdev.littlesnitch.networkextension.systemextension/Contents/_CodeSignature
./at.obdev.littlesnitch.networkextension.systemextension/Contents/_CodeSignature/CodeResources
./at.obdev.littlesnitch.networkextension.systemextension/Contents/MacOS
./at.obdev.littlesnitch.networkextension.systemextension/Contents/MacOS/at.obdev.littlesnitch.networkextension
./at.obdev.littlesnitch.networkextension.systemextension/Contents/Library
./at.obdev.littlesnitch.networkextension.systemextension/Contents/Library/LaunchServices
./at.obdev.littlesnitch.networkextension.systemextension/Contents/Library/LaunchServices/at.obdev.littlesnitch.daemon.bundle
./at.obdev.littlesnitch.networkextension.systemextension/Contents/Library/LaunchServices/at.obdev.littlesnitch.daemon.bundle/Contents
```

```
oompa@MBP-M1 macfuse.fs % pwd
/Library/StagedExtensions/Library/Filesystems/macfuse.fs
oompa@MBP-M1 macfuse.fs % find .
.
./Contents
./Contents/Extensions
./Contents/Extensions/11
./Contents/Extensions/11/macfuse.kext
./Contents/Extensions/11/macfuse.kext/Contents
./Contents/Extensions/11/macfuse.kext/Contents/_CodeSignature
./Contents/Extensions/11/macfuse.kext/Contents/_CodeSignature/CodeResources
./Contents/Extensions/11/macfuse.kext/Contents/MacOS
./Contents/Extensions/11/macfuse.kext/Contents/MacOS/macfuse
./Contents/Extensions/11/macfuse.kext/Contents/Info.plist
```

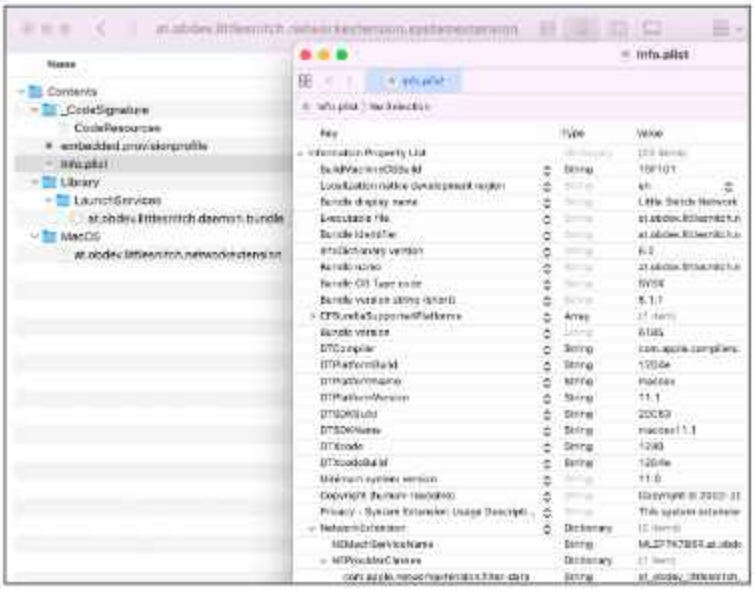
SANS DFIR | FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 67

These examples show a user installed system extension (top) for the Little Snitch network monitor, and a kernel extension (bottom) for the filesystem driver macFUSE.



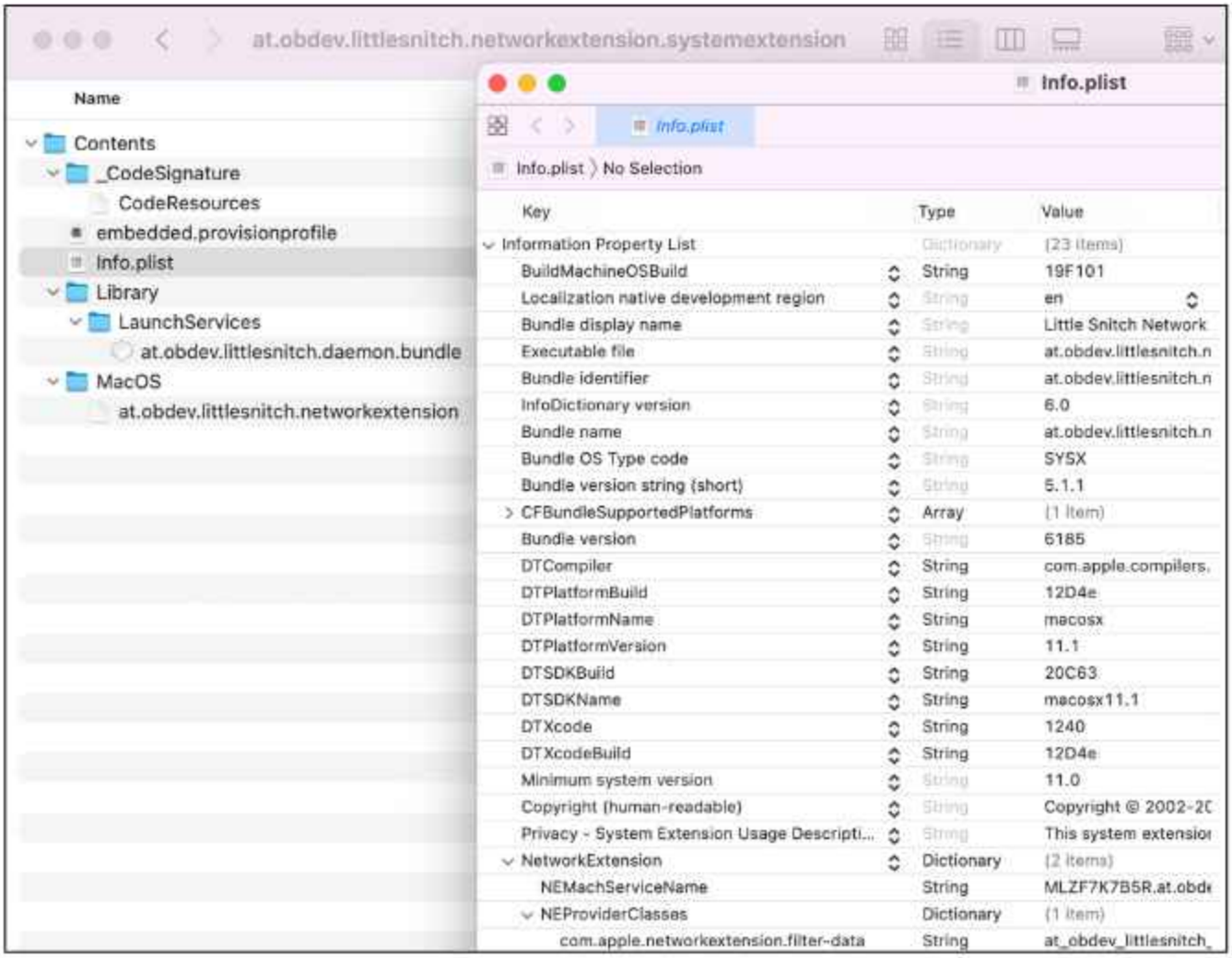
If you are able to perform commands on a live system, these two are highly suggested to list the loaded extensions on the system.

Kernel and System Extensions: Bundle



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

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

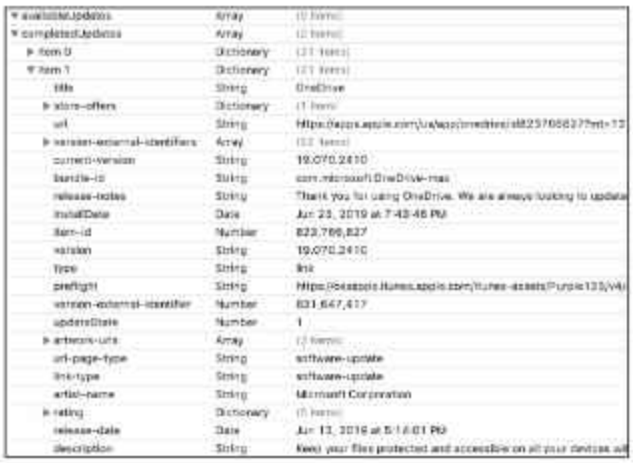


The Apple menu contains the item "`Software Update...`". When selected, it opens the Mac App Store on the "`Updates`" tab, as shown in the screenshot above. This window shows the updates that are available, and both the Apple system updates and any applications that had been downloaded and installed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

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

macOS Software Updates:
System Updates: /Library/Preferences/com.apple.SoftwareUpdate.plist
3rd Party: ~/Library/Caches/com.apple.appstoreagent/updates.plist (10.14) or storeSystem.db (10.15+)

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



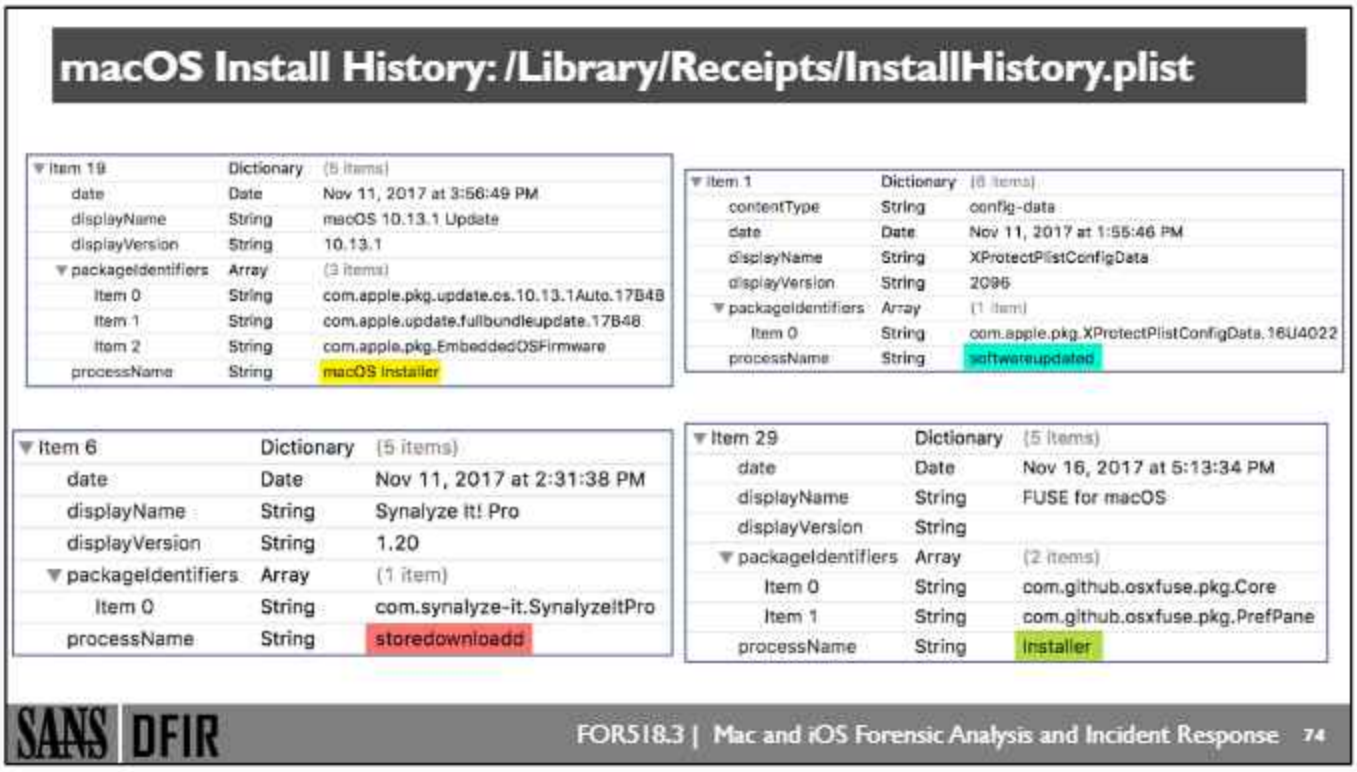
The `com.appleSoftwareUpdate.plist` property list for 10.8+ systems contains additional data. 10.14 introduced a separate plist for third-party updates, updates.plist in the User preferences.

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

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

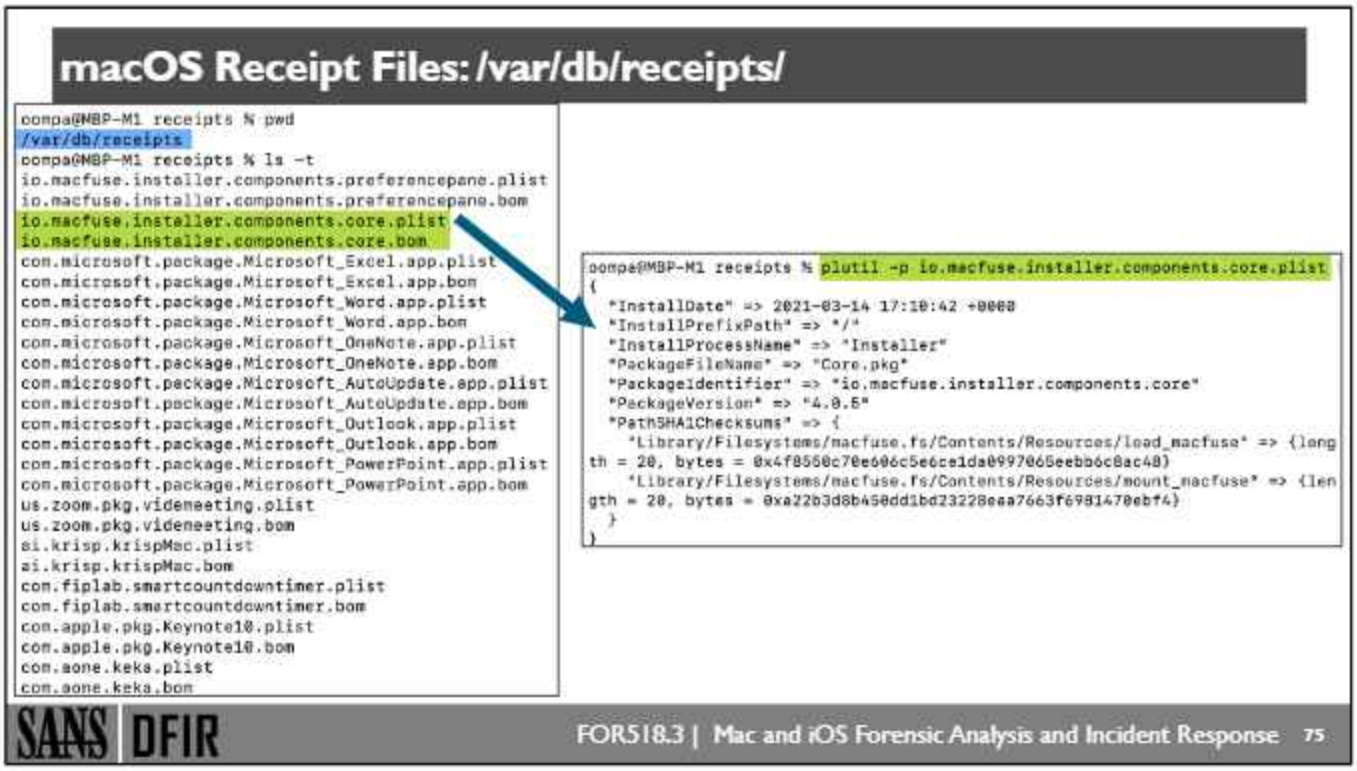
LastResultCode	Number	0
LastAttemptSystemVersion	String	10.14.4 (18E226)
SkipLocalCDN	Boolean	NO
LastUpdatesAvailable	Number	2
LastRecommendedUpdatesAvailable	Number	2
LastAttemptBuildVersion	String	10.14.4 (18E226)
▼ RecommendedUpdates	Array	(2 items)
▼ Item 0	Dictionary	(4 items)
Identifier	String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

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



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

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



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

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

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

```
compa@MBP-M1 receipts % lsbom io.macfuse.installer.components.core.bom
.       40755   0/0
./Library       40755   0/0
./Library/Filesystems   40755   0/0
./Library/Filesystems/macfuse.fs        40755   0/0
./Library/Filesystems/macfuse.fs/Contents       40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions    40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.10      120755  0/0     4       1629272622      10.9
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11      40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext 40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents        40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/Info.plist     100644  0/0     1522    757289844
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/MacOS  40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/MacOS/macfuse  100755  0/0     139760  3792227545
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/_CodeSignature 40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/_CodeSignature/CodeResources   100644  0/0     2280    345647378
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12      40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext 40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents        40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/Info.plist     100644  0/0     1523    1128989417
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/MacOS  40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/MacOS/macfuse  100755  0/0     139984  2387771134
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/_CodeSignature 40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/_CodeSignature/CodeResources   100644  0/0     2280    345647378
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.13      120755  0/0     5       3373115563      10.12
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.14      120755  0/0     5       3373115563      10.12
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.15      120755  0/0     5       3373115563      10.12
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.16      120755  0/0     2       869358703       11
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.9       40755   0/0
```


The (partial) screenshot above shows an example of the output from the command `lsbom`. This shows the various files associated with the BOM file. The full screenshot on the next page shows everything that is installed for macFUSE core components. This makes a good list of places to look to determine if this software is legitimate or not.

Default output contains:

- File path
- Mode
- UID/GID

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

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

Reference:
Man Page for `lsbom`

```
dde@dde$MSP-RI receipts % lsbom io.macfuse.installer.components.core.bom
.       40755   0/0
./Library       40755   0/0
./Library/Filesystems   40755   0/0
./Library/Filesystems/macfuse.fs        40755   0/0
./Library/Filesystems/macfuse.fs/Contents       40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions    40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.10      120755  0/0     4       1629272622      10.9
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11      40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext 40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents        40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/Info.plist     100644  0/0     1522    757209844
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/MacOS  40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/MacOS/macfuse  100755  0/0     139768  3792227645
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/_CodeSignature 40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.11/macfuse.kext/Contents/_CodeSignature/CodeResources   100644  0/0     2200    345647378
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12      40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext 40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents        40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/Info.plist     100644  0/0     1523    1128989417
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/MacOS  40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/MacOS/macfuse  100755  0/0     139984  2387771134
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/_CodeSignature 40755   0/0
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.12/macfuse.kext/Contents/_CodeSignature/CodeResources   100644  0/0     2200    345647378
./Library/Filesystems/macfuse.fs/Contents/Extensions/10.13      120755  0/0     5       3173115563      10.12
./Library/Filesystems/macfuse.fs/Contents/_CodeSignature/CodeRequirements-1     100644  0/0     225     1868851531
./Library/Filesystems/macfuse.fs/Contents/_CodeSignature/CodeResources  100644  0/0     13059   2254285417
./Library/Filesystems/macfuse.fs/Contents/_CodeSignature/CodeSignature  100644  0/0     9853    390969637
./Library/Filesystems/macfuse.fs/Contents/version.plist 100644  0/0     448     11A5093631
./Library/Frameworks    40755   0/0
./Library/Frameworks/OSXFUSE.framework  40755   0/0
./Library/Frameworks/OSXFUSE.framework/Versions 40755   0/0
./Library/Frameworks/OSXFUSE.framework/Versions/A       40755   0/0
./Library/Frameworks/OSXFUSE.framework/Versions/A/OSXFUSE       120755  0/0     56      947896319       /Library/Frameworks/macFUSE.framework/Versions/A/macFUSE
./Library/Frameworks/macFUSE.framework  40755   0/0
./Library/Frameworks/macFUSE.framework/Headers  120755  0/0     24      2473586787      Versions/Current/Headers
./Library/Frameworks/macFUSE.framework/Resources        120755  0/0     26      3382263027      Versions/Current/Resources
./Library/Frameworks/macFUSE.framework/Versions 40755   0/0
./Library/Frameworks/macFUSE.framework/Versions/A       40755   0/0
./Library/Frameworks/macFUSE.framework/Versions/A/Headers        40755   0/0
./Library/Frameworks/macFUSE.framework/Versions/A/Headers/GMAvailability.h      100644  0/0     1463    3436060997
./Library/Frameworks/macFUSE.framework/Versions/A/Headers/GMFinderInfo.h        100644  0/0     3767    3212867956
./Library/Frameworks/macFUSE.framework/Versions/A/Headers/GMResourceFork.h      100644  0/0     5756    734686040
./Library/Frameworks/macFUSE.framework/Versions/A/Headers/GMUserFileSystem.h    100644  0/0     38147   1764988887
./Library/Frameworks/macFUSE.framework/Versions/A/Headers/macFUSE.h     100644  0/0     2813    23343604
./Library/Frameworks/macFUSE.framework/Versions/A/Resources      40755   0/0
./Library/Frameworks/macFUSE.framework/Versions/A/Resources/Info.plist  100644  0/0     1418    1619118208
./Library/Frameworks/macFUSE.framework/Versions/A/_CodeSignature        40755   0/0
./Library/Frameworks/macFUSE.framework/Versions/A/_CodeSignature/CodeResources  100644  0/0     3597    834320890
./Library/Frameworks/macFUSE.framework/Versions/A/macFUSE       100755  0/0     257968  2163794691
./Library/Frameworks/macFUSE.framework/Versions/Current 120755  0/0     1       1751287896      A
./Library/Frameworks/macFUSE.framework/macFUSE  120755  0/0     24      2438888327      Versions/Current/macFUSE
./usr   40755   0/0
./usr/local     40755   0/0
./usr/local/include     40755   0/0
./usr/local/include/fuse        40755   0/0
./usr/local/include/fuse/fuse.h 100644  0/0     37962   3243410887
./usr/local/include/fuse/fuse_common.h  100644  0/0     15272   1067492118
./usr/local/include/fuse/fuse_common_compat.h   100644  0/0     714     3825354545
./usr/local/include/fuse/fuse_compat.h  100644  0/0     8129    468315287
./usr/local/include/fuse/fuse_lowlevel.h        100644  0/0     56319   1870999744
./usr/local/include/fuse/fuse_lowlevel_compat.h 100644  0/0     6929    866895299
./usr/local/include/fuse/fuse_opt.h     100644  0/0     7477    336153515
./usr/local/include/fuse.h      100644  0/0     246     2897683619
./usr/local/lib 40755   0/0
./usr/local/lib/libfuse.2.dylib 100755  0/0     580688  3225699622
./usr/local/lib/libfuse.dylib   120755  0/0     15      2789138727      libfuse.2.dylib
./usr/local/lib/libfuse.la      100755  0/0     971     1784680650
./usr/local/lib/libosxfuse.2.dylib      120755  0/0     15      2789138727      libfuse.2.dylib
./usr/local/lib/libosxfuse_i64.2.dylib  120755  0/0     15      2789138727      libfuse.2.dylib
./usr/local/lib/pkgconfig       40755   0/0
./usr/local/lib/pkgconfig/fuse.pc       100644  0/0     321     2561190528
```

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

```
oompa@Sarahs-MBA ~ % grep Installed /var/log/install.log
2020-06-17 19:57:45-04 Sarahs-MBA installd[814]: Installed "Microsoft Word" ()
2020-06-17 20:05:53-04 Sarahs-MBA installd[814]: Installed "Microsoft Excel" ()
2020-06-17 20:17:45-04 Sarahs-MBA installd[814]: Installed "Microsoft PowerPoint" ()
2020-06-18 16:16:53-04 Sarahs-MBA installd[814]: Installed "Microsoft AutoUpdate" ()
2020-06-22 18:04:19-04 Sarahs-MBA installd[814]: Installed "Developer App" (8.2.0.0)
2020-07-03 10:14:05-04 Sarahs-MBA installd[814]: Installed "AS Timer" (5.2)
2020-07-03 10:14:24-04 Sarahs-MBA installd[814]: Installed "Telegram" (6.2.2)
2020-07-03 10:14:31-04 Sarahs-MBA installd[814]: Installed "Snipposé" (1.8)
2020-07-03 10:14:36-04 Sarahs-MBA installd[814]: Installed "Developer App" (8.2.0.0)
2020-07-03 10:15:51-04 Sarahs-MBA installd[814]: Installed "WhatsApp" (2.2025.7)
2020-07-03 10:16:11-04 Sarahs-MBA installd[814]: Installed "Keka" (1.1.30)
```


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

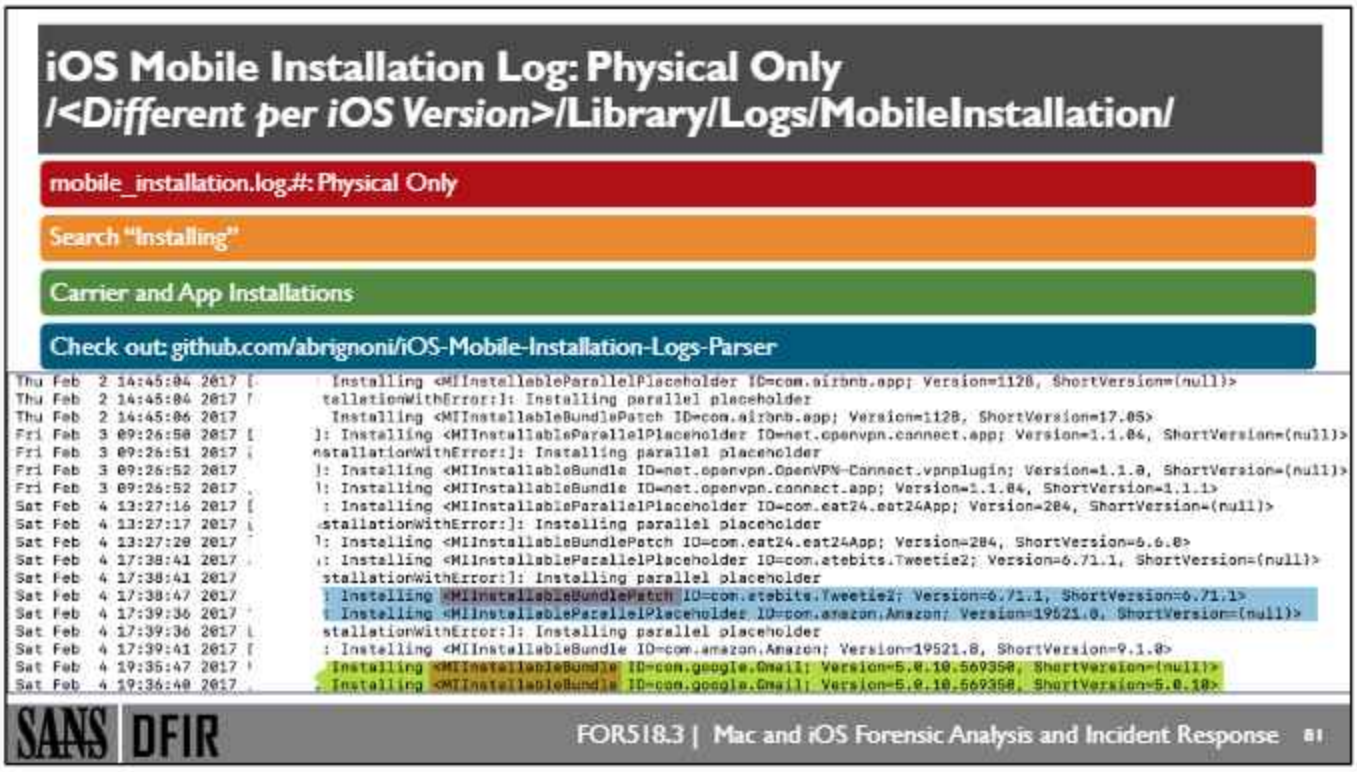
We can look at the `install.log` on the system and search for the keyword "`Installed`" to easily show the software packages installed. This log does not include installation information of command-line tools such as `homebrew`.

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

Install Details: /var/log/install.log

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

- Where the software package was opened from on disk.
- Code Signing verification.
- Receipt creation.
- Was administrator authorization needed? – "Administrator authorization granted" would show in the logs (not in the example install).



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

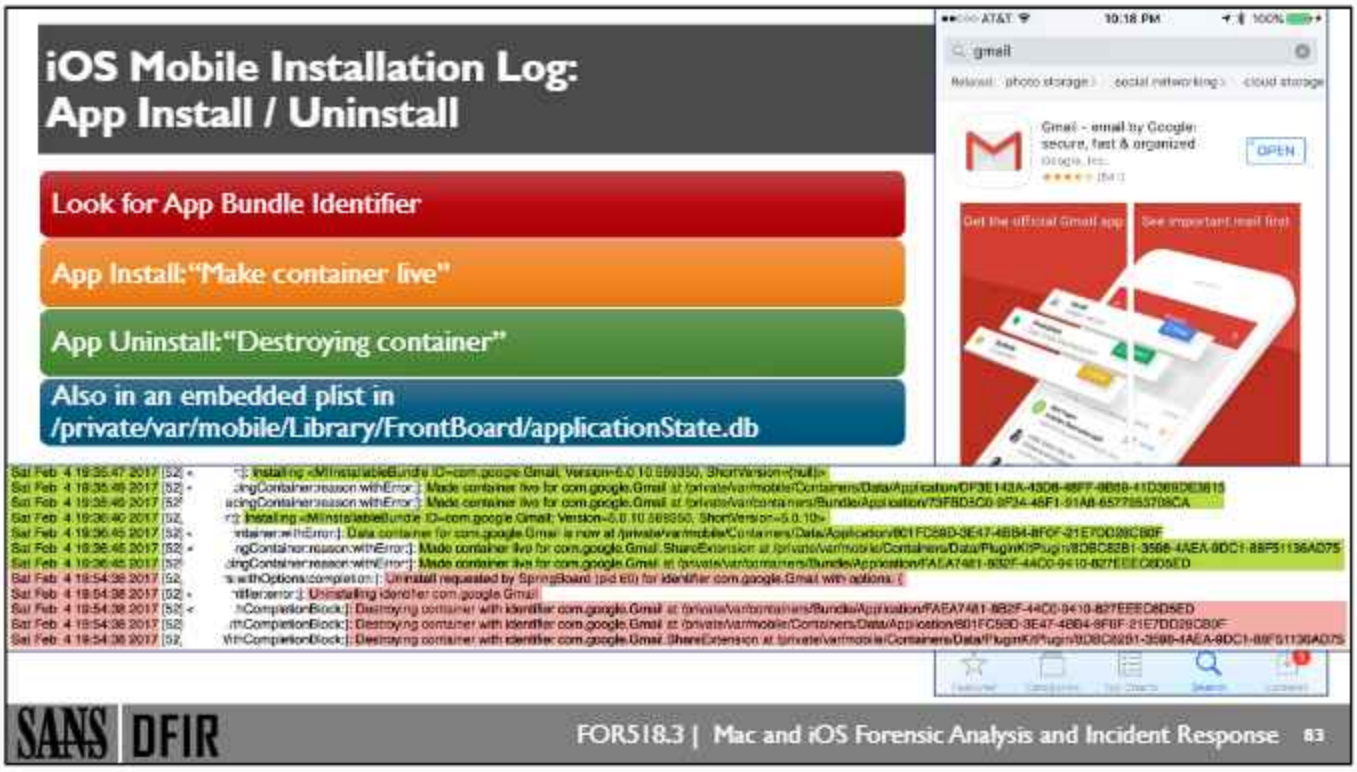
The `mobile_installation.log.#` log files located in the `/MobileInstallation/` directory contains a historical view of when apps were installed, and carrier bundles were installed. How long these entries are kept depends on the version of iOS being used. The older the iOS version, the longer these entries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

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

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

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

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



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

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

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

```
Sat Feb  4 19:35:47 2017 [52] <
Sat Feb  4 19:35:49 2017 [52] <
Sat Feb  4 19:35:49 2017 [52]
Sat Feb  4 19:36:40 2017 [52]
Sat Feb  4 19:36:45 2017 [52] <
Sat Feb  4 19:36:45 2017 [52] <
Sat Feb  4 19:36:45 2017 [52]
Sat Feb  4 19:54:38 2017 [52]
Sat Feb  4 19:54:38 2017 [52] <
Sat Feb  4 19:54:38 2017 [52]
Sat Feb  4 19:54:38 2017 [52]
Sat Feb  4 19:54:38 2017 [52]
```

```
*]: Installing <MIInstallableBundle ID=com.google.Gmail; Version=5.0.10.569350; ShortVersion=(null)>
JingContainer:reason:withError:]: Made container live for com.google.Gmail at /private/var/mobile/Containers/Data/Application/DF3E143A-43D8-48FF-9B59-41D389DE3815
acingContainer:reason:withError:]: Made container live for com.google.Gmail at /private/var/containers/Bundle/Application/73FB05C0-9F34-45F1-91A8-8577953708CA
r]: Installing <MIInstallableBundle ID=com.google.Gmail; Version=5.0.10.569350; ShortVersion=5.0.10>
ntainer:withError:]: Data container for com.google.Gmail is now at /private/var/mobile/Containers/Data/Application/801FC59D-3E47-4BB4-8F0F-21E7D028C80F
ngContainer:reason:withError:]: Made container live for com.google.Gmail.ShareExtension at /private/var/mobile/Containers/Data/PluginKitPlugin/BDBC82B1-3598-4AEA-9DC1-88F51136AD75
JingContainer:reason:withError:]: Made container live for com.google.Gmail at /private/var/containers/Bundle/Application/FAEA7481-8B2F-44C0-9410-827EEEC6D5ED
's:withOptions:completion:]: Uninstall requested by SpringBoard (pid 60) for identifier com.google.Gmail with options: {
ntifier:error:]: Uninstalling identifier com.google.Gmail
:]:CompletionBlock:]: Destroying container with identifier com.google.Gmail at /private/var/containers/Bundle/Application/FAEA7481-8B2F-44C0-9410-827EEEC6D5ED
:]:CompletionBlock:]: Destroying container with identifier com.google.Gmail at /private/var/mobile/Containers/Data/Application/801FC59D-3E47-4BB4-8F0F-21E7D028C80F
WithCompletionBlock:]: Destroying container with identifier com.google.Gmail.ShareExtension at /private/var/mobile/Containers/Data/PluginKitPlugin/BDBC82B1-3598-4AEA-9DC1-88F51136AD75
```

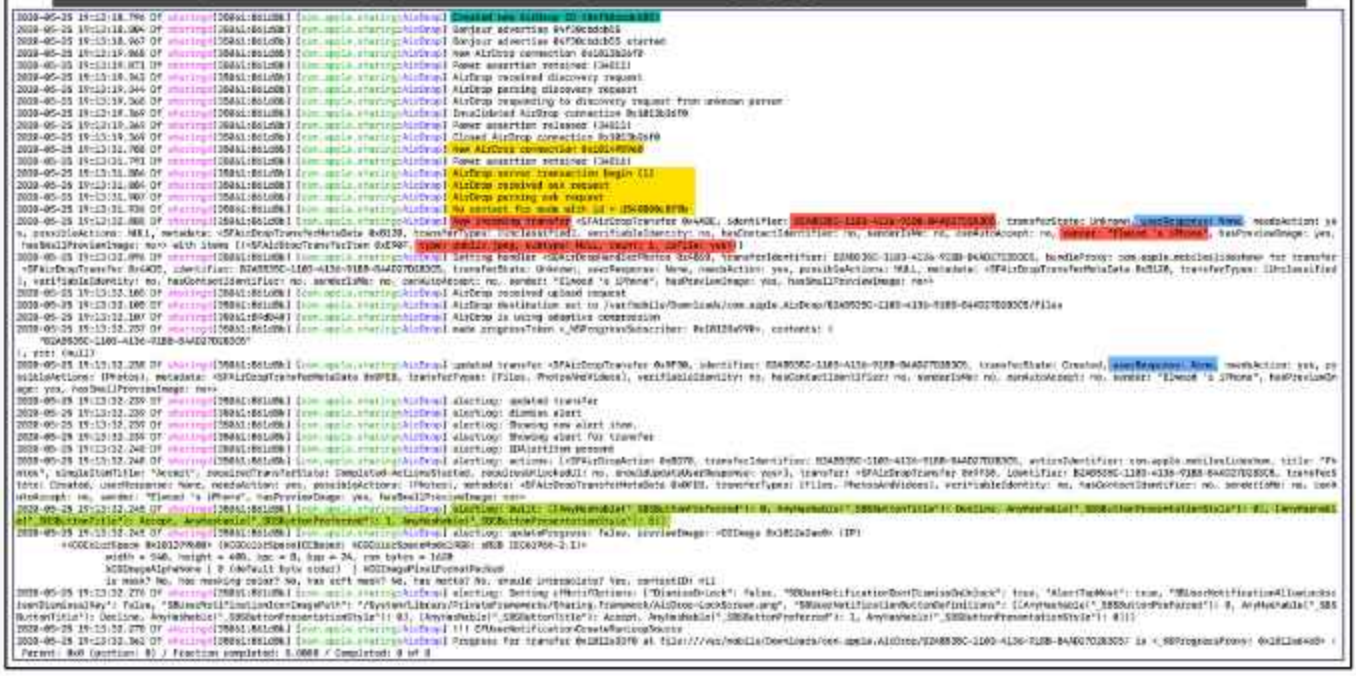
AirDrop Activity (Sending from iOS)

AirDrop activity is also tracked in Unified Logs. This example shows an iPhone sending a photo to another iPhone (miPhone11).

If both devices (macOS or iOS) can be reviewed, this activity can be tied together with the AirDrop ID (ReceiverID). If only one device is available, this attribution becomes far more difficult.

Reference:
https://www.mac4n6.com/blog/2020/6/5/analysis-of-apple-unified-logs-quarantine-edition-entry-11-airdropping-some-knowledge

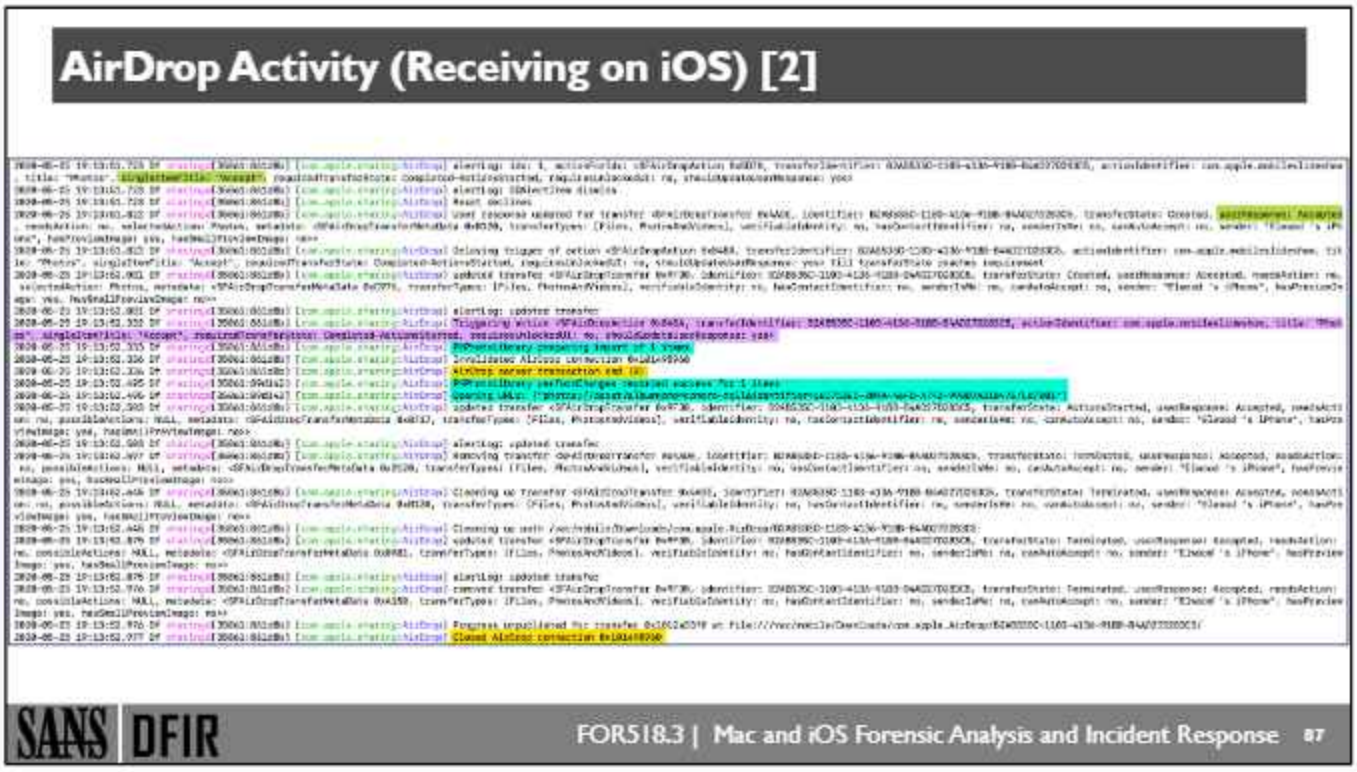
AirDrop Activity (Receiving on iOS) [1]



The receiving side of this AirDrop is shown above from miPhone11. It shows that it came from 'Elwood's iPhone', but a device hostname is easily changed.

This first part show the transaction information including the AirDrop IDs, file type being sent, and the alert window presented to the user.

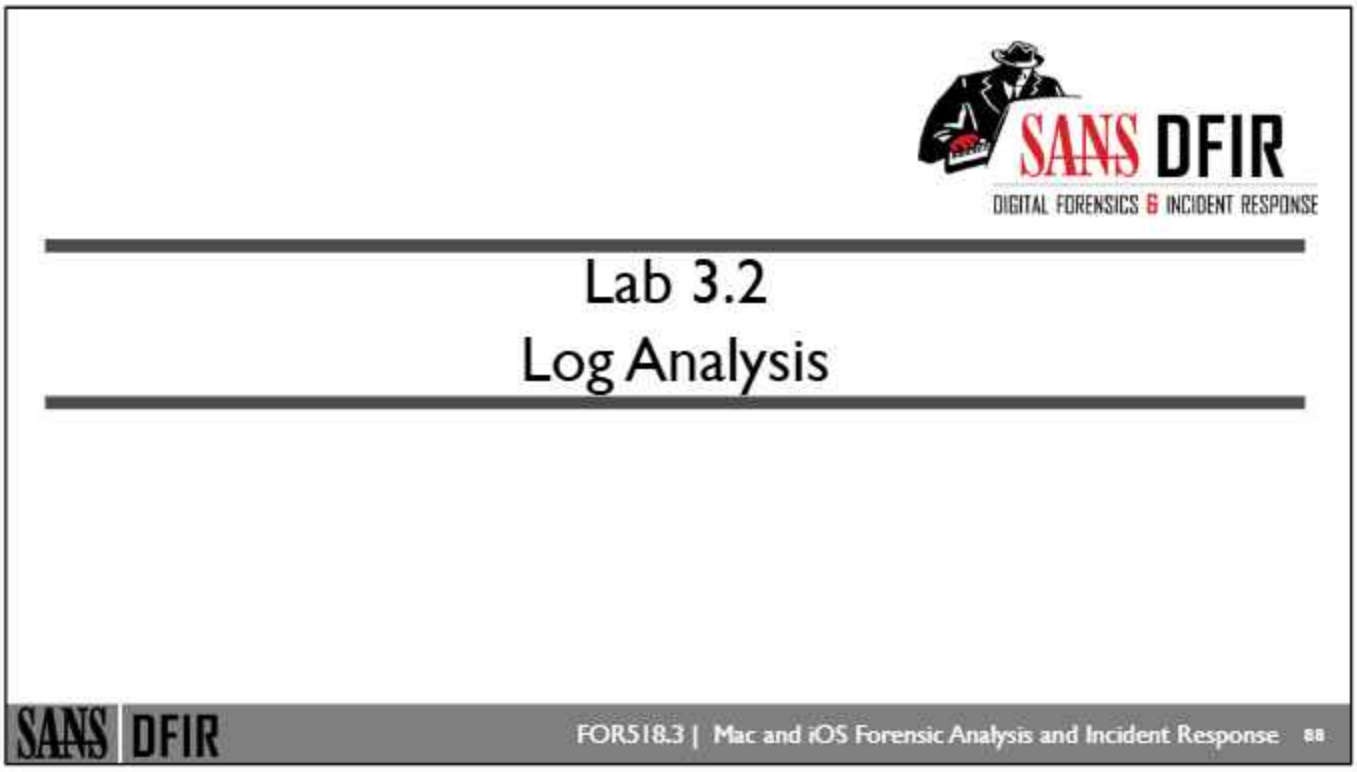
Reference:
https://www.mac4n6.com/blog/2020/6/5/analysis-of-apple-unified-logs-quarantine-edition-entry-11-airdropping-some-knowledge



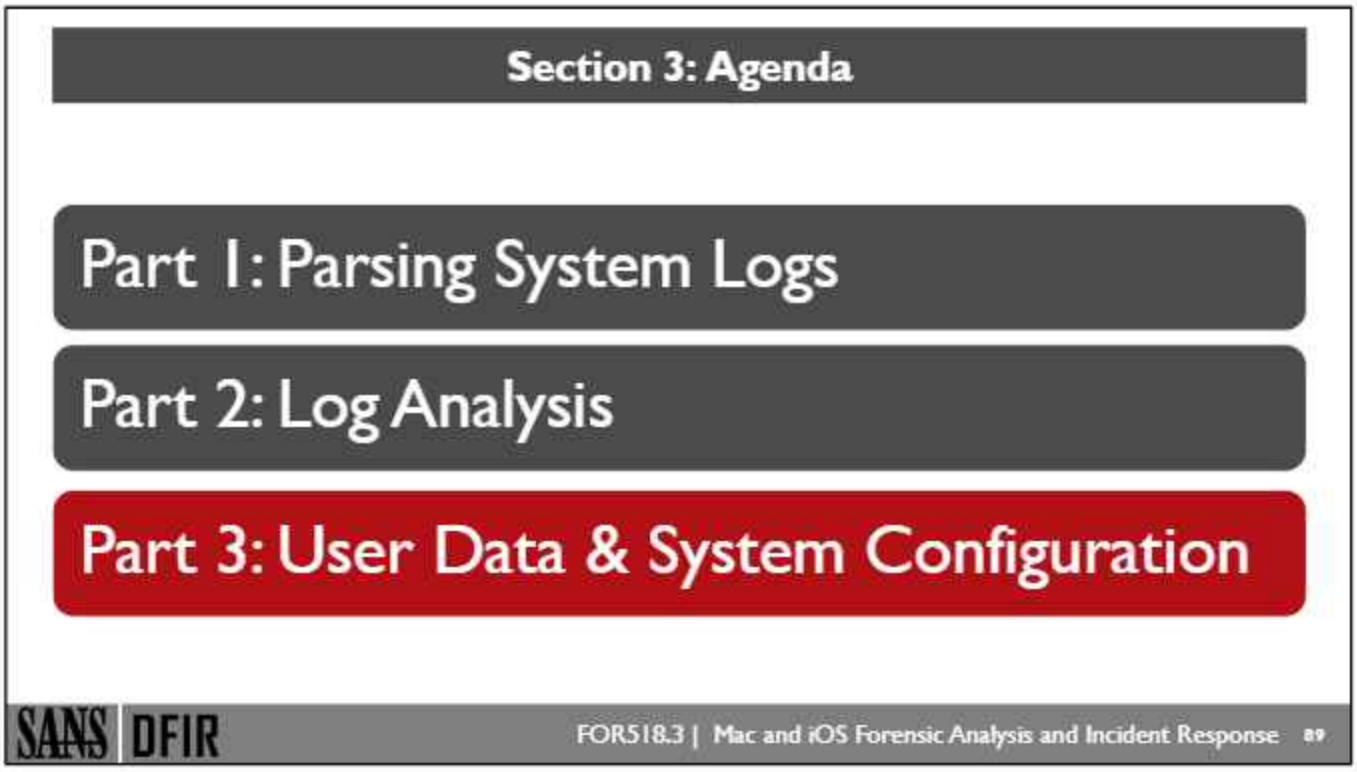
The receiving side of this AirDrop is shown above from miPhone11. It shows that it came from 'Elwood's iPhone' but a device hostname is easily changed.

This second part shows that the connection was 'Accepted' (versus Declined) and that the photo was imported into the Photos app.

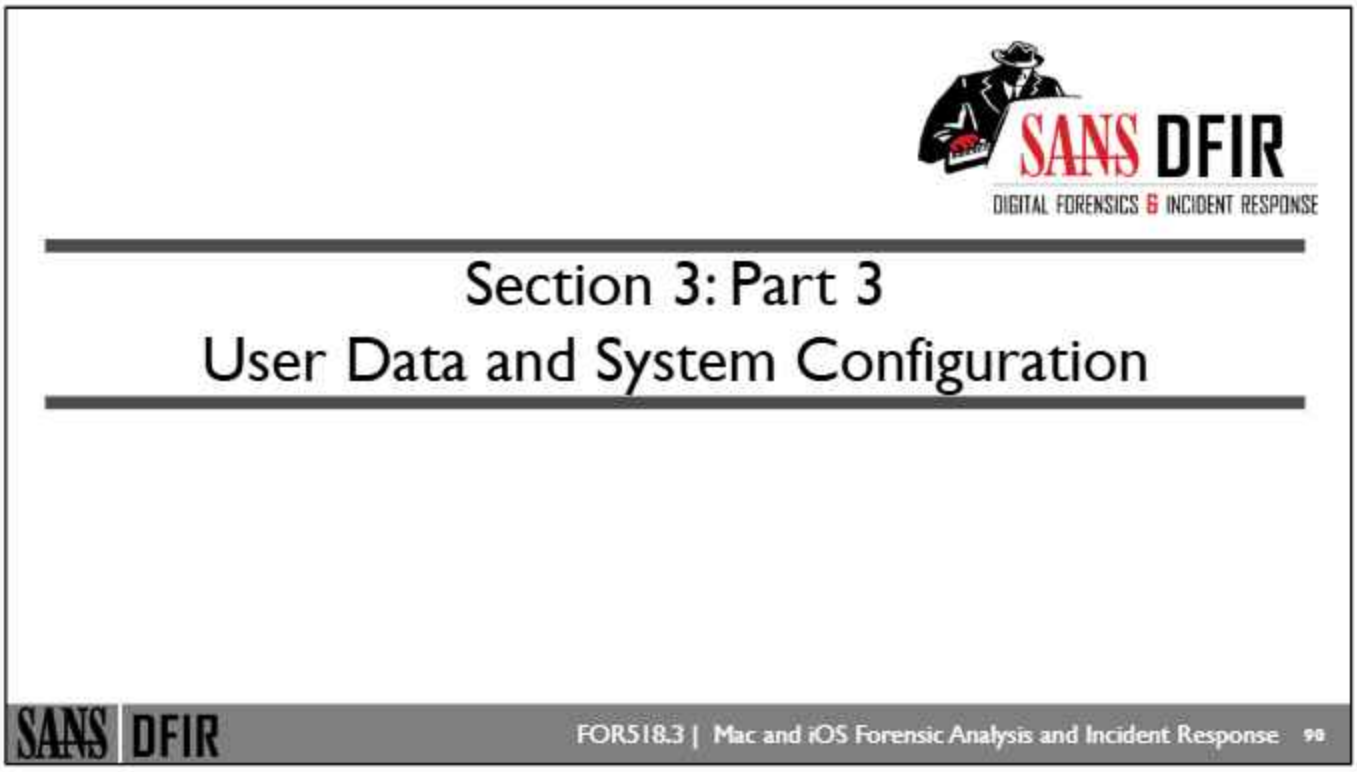
Reference:
https://www.mac4n6.com/blog/2020/6/5/analysis-of-apple-unified-logs-quarantine-edition-entry-11-airdropping-some-knowledge



This page intentionally left blank.



This page intentionally left blank.



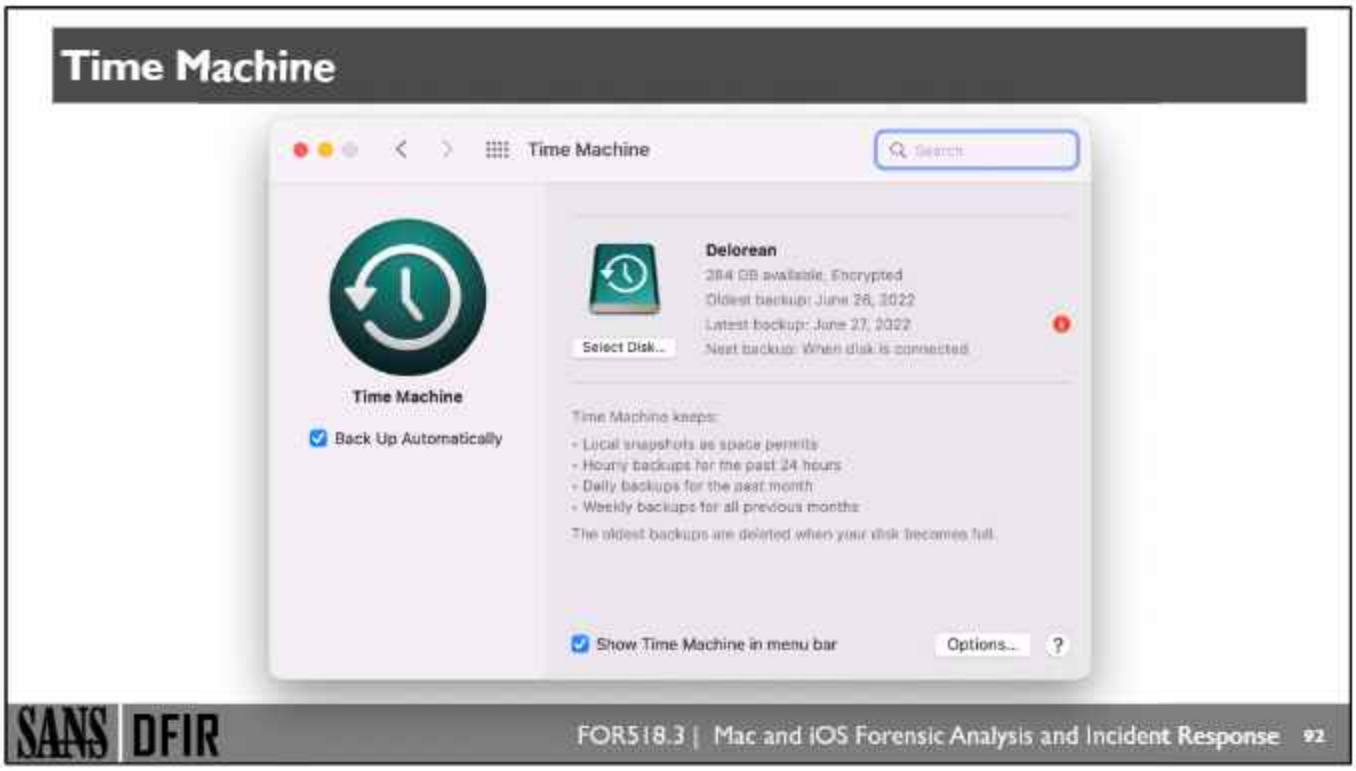
This page intentionally left blank.

User & System Configuration – Preference Panels

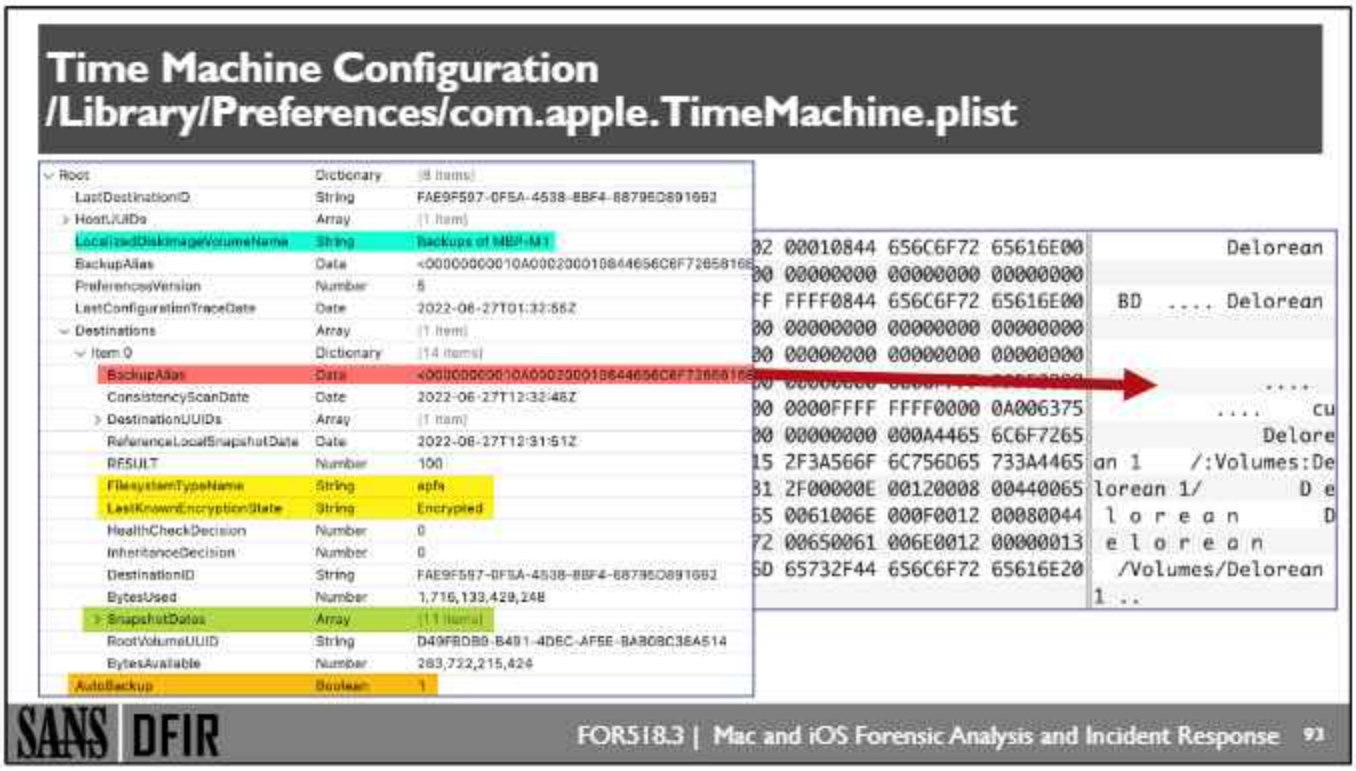
Many of the user and system configuration can be found in the System Preferences window or in Settings on iOS.

In the next sections we will discuss many of these configurations as they pertain to configuration storage files and some items that can be found in various logs.

On macOS systems each of these setting options bring up what is called a "Preference Panel" to configure the settings for that item. (i.e., the Security & Privacy preference panel).



The Time Machine Preference Panel can be used to configure a network or local backup disk to acquire backups of the Mac system.



Time Machine is the native backup utility on macOS systems. The settings plist can be found in the system preferences directory - `/Library/Preferences/com.apple.TimeMachine.plist`.

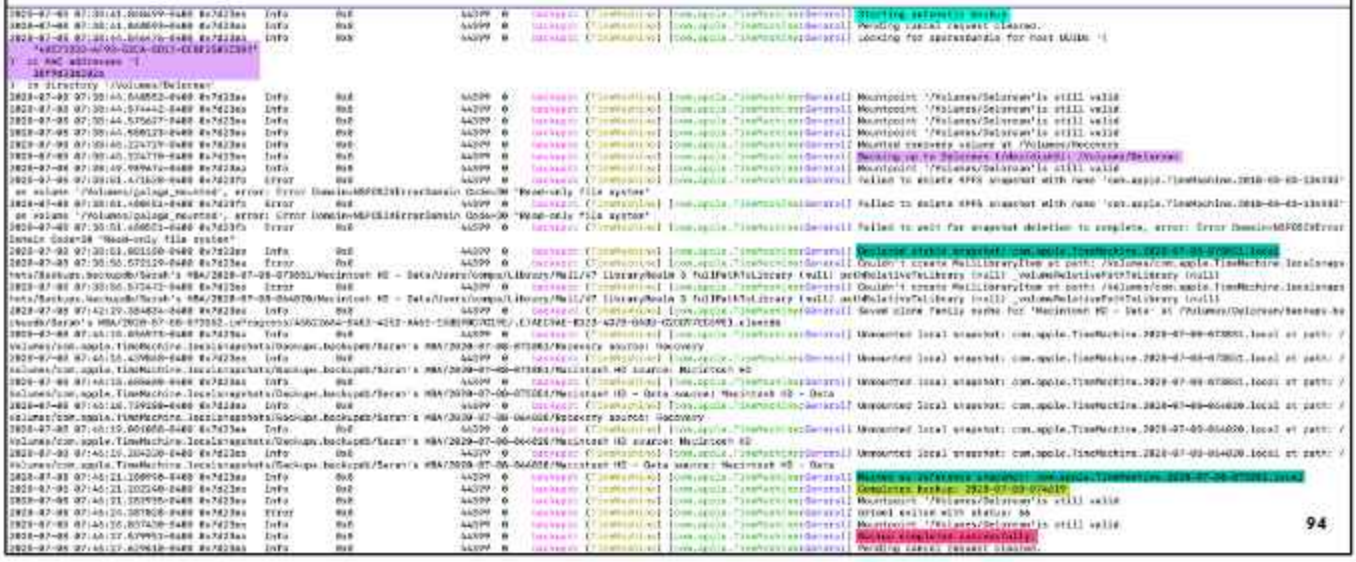
This plist contains information such as `BackupAlias` that has a bookmark blob with device information such as the name of the backup disk (mine is named Delorean).

There are also timestamps associated with the backup snapshots in `SnapshotDates`.

Other configuration items such as the file system (apfs), encryption status, and backup frequency (`AutoBackup`) can be found in this plist file.

Time Machine – "backupd" Process

```
log show --info --predicate 'process = "backupd" and
category = "General"'
```



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

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

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

When just the term 'backupd' is used for the predicate, a few more categories are shown that include information on:

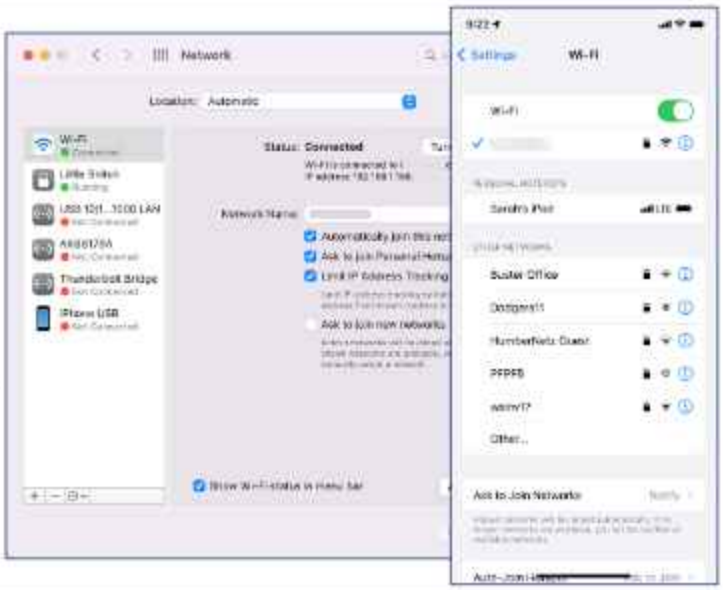
- Determining sizing requirements
- Copy progression
- Event Collections
- Etc.

Even a stray filename may appear in this output as it determines what needs to be backed up! In a following slide an example of `log show --info --predicate 'process = "backupd"'` is shown.

```
2020-07-08 07:38:52.683 I  backupd[44399:7d23ea] [com.apple.TimeMachine:CloneCollection] Read 2942 clone families from /Volumes/Delorean/Backups.backupdb/Sarah's MBA/2020-07-08-064503/.E7AEC9AE-8323-4D7B-8AB3-C2CC97EC6951.clonedb
2020-07-08 07:38:52.698 I  backupd[44399:7d23ea] [com.apple.TimeMachine:EventCollection] Buffering events...
2020-07-08 07:38:52.753 I  backupd[44399:7d23ea] [com.apple.TimeMachine:EventCollection] Buffered 2713 events.
2020-07-08 07:38:52.756 I  backupd[44399:7d23ea] [com.apple.TimeMachine:EventCollection] Collected 1 events. Last path seen: 'private/var/folders/zz/zyxvpxxqdcsfxvn_n0000000000000/Cleanup At Startup/com.apple.backupd.filechecklist'
2020-07-08 07:38:54.461 I  backupd[44399:7d23ea] [com.apple.TimeMachine:EventCollection] Trusting source modification times for complete event history.
2020-07-08 07:38:54.461 I  backupd[44399:7d23ea] [com.apple.TimeMachine:CloneCollection] Found 3 perfect clone families, 3 partial clone families. 204.5 MB physical space used by clone files. 78.5 MB shared space.
2020-07-08 07:38:54.467 I  backupd[44399:7d23ea] [com.apple.TimeMachine:EventCollection] Saved event cache at /Volumes/Delorean/Backups.backupdb/Sarah's MBA/2020-07-08-073852.inProgress/4B6226A4-8453-4262-AA6E-158890C7C19E/.E7AEC9AE-832
3-4D7B-8AB3-C2CC97EC6951.eventdb
2020-07-08 07:38:54.478 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Estimating size of changes for "Recovery" (device: /dev/disk1s3 mount: '/Volumes/Recovery' fsUUID: 24C27928-076F-48E9-9588-F6F7937A2778 eventDBUUID:
 (null))
2020-07-08 07:38:54.483 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] (fsk:0,dsk:1,fsz:0,dsz:0)(1/0)
2020-07-08 07:38:54.486 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Sizing by file event iteration for source Recovery completed. Changed content estimate: (bytes: 0 (Zero KB), items: 2)
2020-07-08 07:38:54.495 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Estimated 2 files (Zero KB) need to be backed up from 'Recovery'
2020-07-08 07:38:54.500 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Estimating size of changes for "Macintosh HD" (device: /dev/disk1s5 mount: '/' fsUUID: 57D8F56A-EEDC-45FC-A267-60E0D01C8356 eventDBUUID: 7A97D5A8-36
D6-4558-95C1-56833EAB8729)
2020-07-08 07:38:54.504 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Sizing by file event iteration for source Macintosh HD completed. Changed content estimate: (bytes: 0 (Zero KB), items: 1)
2020-07-08 07:38:54.508 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Estimated 1 files (Zero KB) need to be backed up from 'Macintosh HD'
2020-07-08 07:38:54.512 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Estimating size of changes for "Macintosh HD - Data" (device: /dev/disk1s1 mount: '/System/Volumes/Data' fsUUID: E7AEC9AE-8323-4D7B-8AB3-C2CC97EC695
1 eventDBUUID: 93B00DE4-F4A7-4987-890B-318BD2FE2733)
2020-07-08 07:38:54.518 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] (fsk:0,dsk:1,fsz:0,dsz:0)(3/273)
2020-07-08 07:38:56.572 E  backupd[44399:7d23ea] [com.apple.TimeMachine:General] Couldn't create MailLibraryItem at path: /Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/Sarah's MBA/2020-07-08-073851/Macintosh HD - Data/U
sers/oompa/Library/Mail/V7 libraryRealm 3 fullPathToLibrary (null) pathRelativeToLibrary (null) _volumeRelativePathToLibrary (null)
2020-07-08 07:38:56.572 E  backupd[44399:7d23ea] [com.apple.TimeMachine:General] Couldn't create MailLibraryItem at path: /Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/Sarah's MBA/2020-07-08-064820/Macintosh HD - Data/U
sers/oompa/Library/Mail/V7 libraryRealm 3 fullPathToLibrary (null) pathRelativeToLibrary (null) _volumeRelativePathToLibrary (null)
2020-07-08 07:39:02.031 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Sizing by file event iteration for source Macintosh HD - Data completed. Changed content estimate: (bytes: 103271759 (103.3 MB), items: 438)
2020-07-08 07:39:02.036 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Estimated 438 files (103.3 MB) need to be backed up from 'Macintosh HD - Data'
2020-07-08 07:39:02.037 I  backupd[44399:7d23ea] [com.apple.TimeMachine:SizingProgress] Estimated 441 files (103.3 MB) need to be backed up from all sources
2020-07-08 07:39:03.289 I  backupd[44399:7d23ea] [com.apple.TimeMachine:CopyProgress] Copied 1 items (Zero KB) from volume Recovery. Linked 1. Moved 0
2020-07-08 07:39:03.290 I  backupd[44399:7d23ea] [com.apple.TimeMachine:CloneCollection] Found 0 perfect clone families, 0 partial clone families. Shared space Zero KB (0 bytes).
2020-07-08 07:39:03.296 Df backupd[44399:7d2536] [com.apple.CoreAnalytics:client] Received configuration update from daemon (initial)
2020-07-08 07:39:04.688 I  backupd[44399:7d23ea] [com.apple.TimeMachine:CopyProgress] Copied 5 items (59 bytes) from volume Macintosh HD. Linked 7. Moved 0
2020-07-08 07:39:04.690 I  backupd[44399:7d23ea] [com.apple.TimeMachine:CloneCollection] Found 0 perfect clone families, 0 partial clone families. Shared space Zero KB (0 bytes).
2020-07-08 07:42:29.285 I  backupd[44399:7d23ea] [com.apple.TimeMachine:CopyProgress] Copied 795 items (120 MB) from volume Macintosh HD - Data. Linked 14683. Moved 0
2020-07-08 07:42:29.286 I  backupd[44399:7d23ea] [com.apple.TimeMachine:CloneCollection] Found 3 perfect clone families, 3 partial clone families. Shared space 78.5 MB (78509547 bytes).
2020-07-08 07:42:29.384 I  backupd[44399:7d23ea] [com.apple.TimeMachine:General] Saved clone family cache for 'Macintosh HD - Data' at /Volumes/Delorean/Backups.backupdb/Sarah's MBA/2020-07-08-073852.inProgress/4B6226A4-8453-4262-AA6E-1
58890C7C19E/.E7AEC9AE-8323-4D7B-8AB3-C2CC97EC6951.clonedb
2020-07-08 07:42:47.984 I  backupd[44399:7d2a35] [com.apple.TimeMachine:PowerManagement] Thermal pressure level 0 -> 1
```

Network

- Recall:
  - NetworkInterfaces.plist
  - preferences.plist
  - DHCP Lease plists
  - com.apple.wifi.known-networks.plist
    - com.apple.airport.preferences.plist
    - com.apple.wifi.plist
  - com.apple.commcenter.*



The Network Preference Panel can be used to configure different network configurations.

Unified Logs Example: Network Usage

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

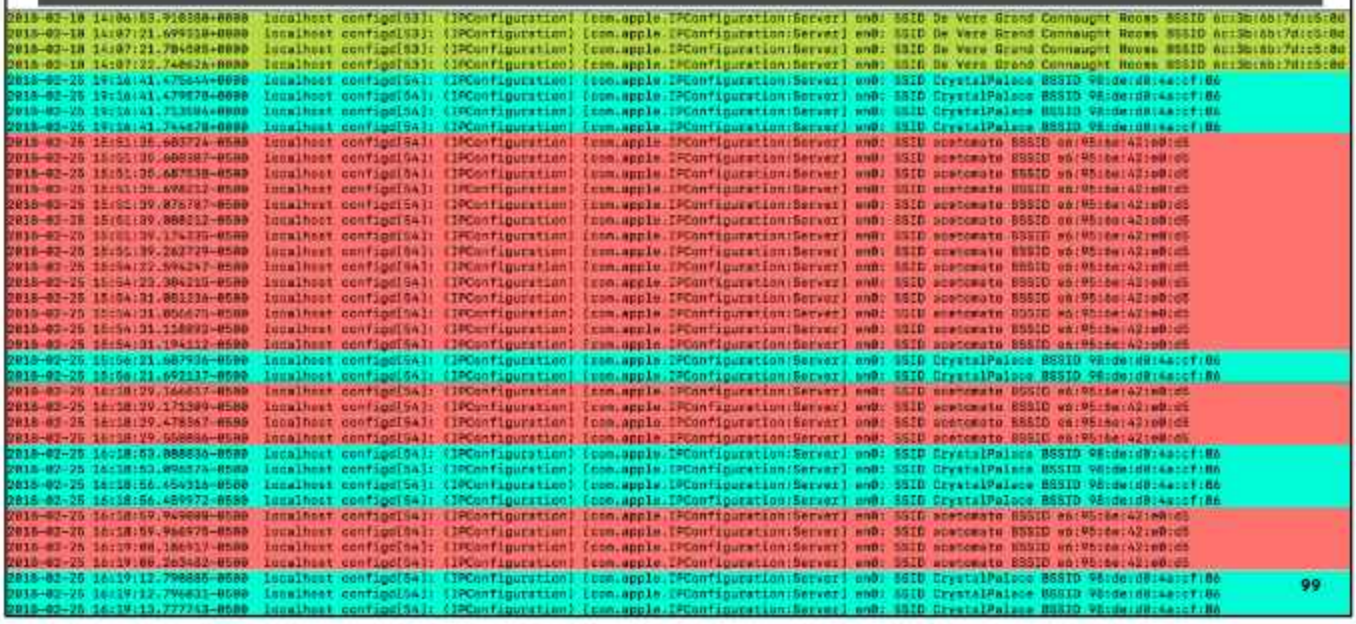
```
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] Renewing Static Lease 172.19.131.137 Router 172.19.131.2
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: no SSID
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: status = 'network changed'
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP bridge0: status = 'network changed'
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:23:ea:7f:4d:91
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: status = 'network changed'
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID is now hhonors (was United_Wi-Fi)
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: No lease for hhonors
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] AUTOMATIC-V6 en0: status = 'network changed'
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:23:ea:7f:4d:91
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:23:ea:7f:4d:91
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:23:ea:7f:6d:91
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stations
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stations
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stations
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] en0: SSID hhonors BSSID 0:24:13:8:7f:21
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stations
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] ignoring lease with SSID stations
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: ARP router: No leases to query for
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: INIT lease = { start 0x1e8efa9e, t1 0x1e8f81a2, t2 0x1e8f85e8, expiration 0x1e8f88aa }
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] DHCP en0: SELECT lease = { start 0x1e8efa9c, t1 0x1e8f81a4, t2 0x1e8f85ea, expiration 0x1e8f88ac }
configd: (IPConfiguration) [com.apple.IPConfiguration.Server] Saved lease for 167.16.207.186
```

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

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

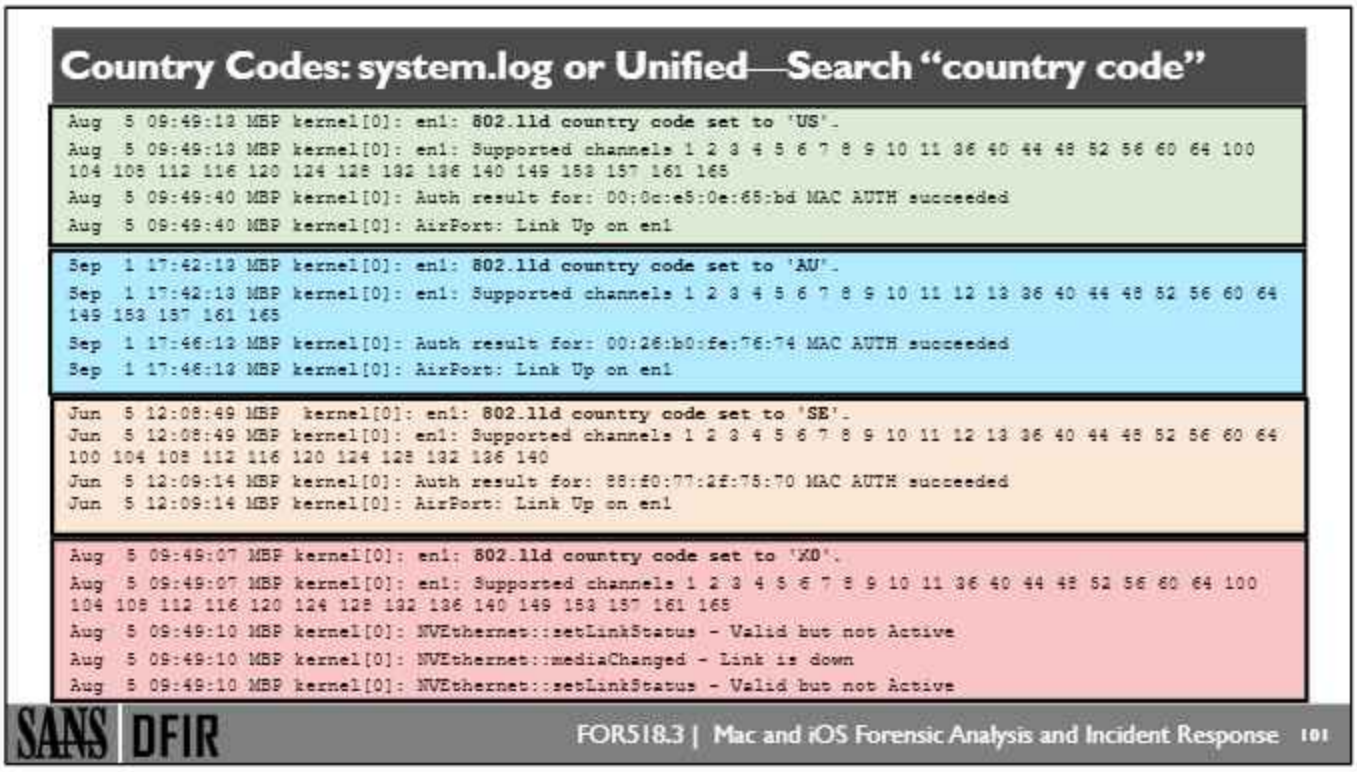
References:
Man Page for `log` Command
https://developer.apple.com/documentation/os/logging
https://developer.apple.com/videos/play/wwdc2016/721/

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



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

```
2018-02-10 14:06:53.910308+0000 localhost configd[53]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID De Vere Grand Connaught Rooms BSSID 6c:3b:6b:7d:c5:8d
2018-02-10 14:07:21.699319+0000 localhost configd[53]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID De Vere Grand Connaught Rooms BSSID 6c:3b:6b:7d:c5:8d
2018-02-10 14:07:21.704505+0000 localhost configd[53]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID De Vere Grand Connaught Rooms BSSID 6c:3b:6b:7d:c5:8d
2018-02-10 14:07:22.740620+0000 localhost configd[53]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID De Vere Grand Connaught Rooms BSSID 6c:3b:6b:7d:c5:8d
2018-02-25 19:16:41.475644+0000 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 19:16:41.479278+0000 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 19:16:41.713584+0000 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 19:16:41.744678+0000 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 15:51:35.603724-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:35.608307-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:35.657538-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:35.698212-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:39.876787-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:39.880212-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:39.174335-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:51:39.262720-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:22.594247-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:23.304215-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:31.851238-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:31.856675-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:31.118892-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:54:31.194112-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 15:56:21.687936-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 15:56:21.692137-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:29.166817-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:18:29.558886-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:18:53.880636-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:53.896574-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:56.454316-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:56.459972-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:18:59.940080-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:18:59.966976-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:19:00.186917-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:19:00.263482-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID acetomato BSSID e6:95:6e:42:e0:d5
2018-02-25 16:19:12.790885-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:19:12.796031-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
2018-02-25 16:19:13.777743-0500 localhost configd[54]: [IPConfiguration] [com.apple.IPConfiguration:Server] en0: SSID CrystalPalace BSSID 98:de:d0:4a:cf:86
```



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

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

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

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

GUI Artifacts

- macOS • Finder
- iOS • SpringBoard
- watchOS • Carousel*
- tvOS • PineBoard*

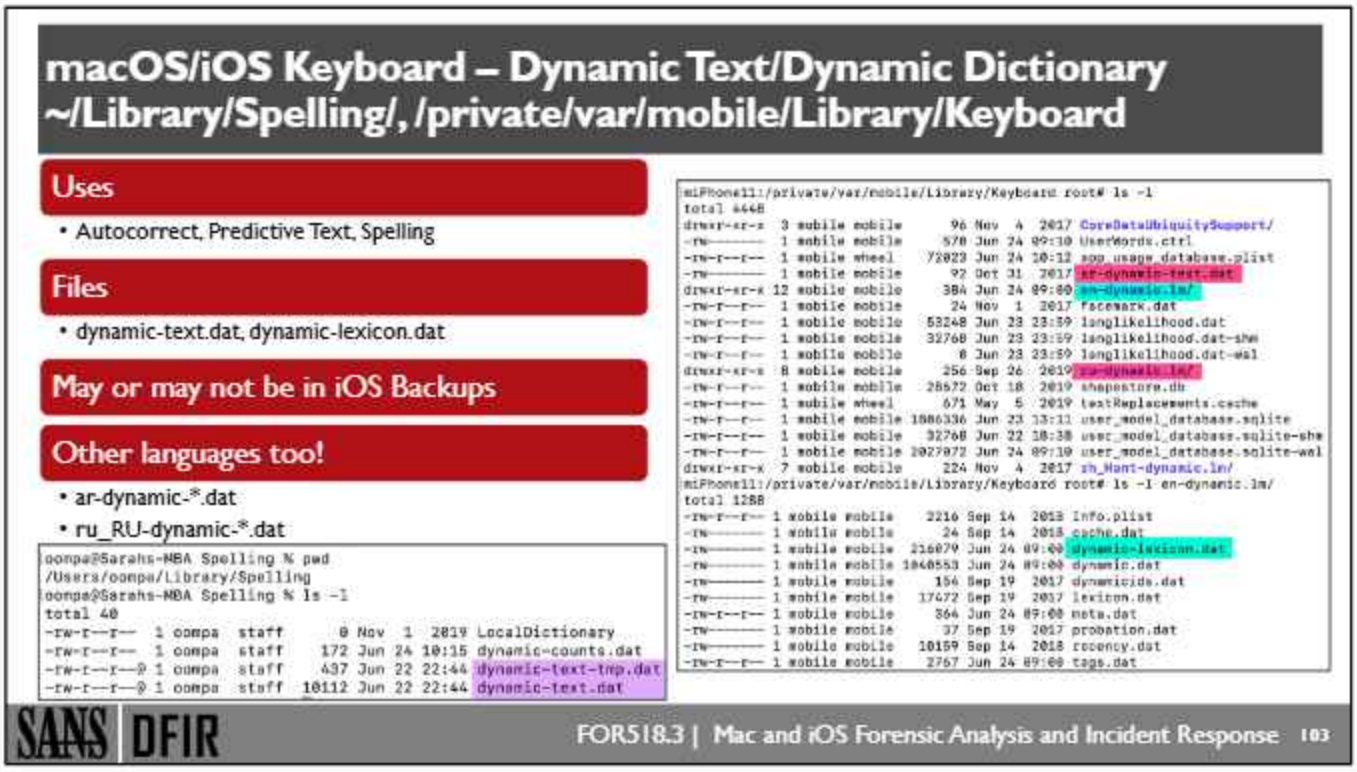
*Not covered, but good bar trivia

SANS DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 102

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

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

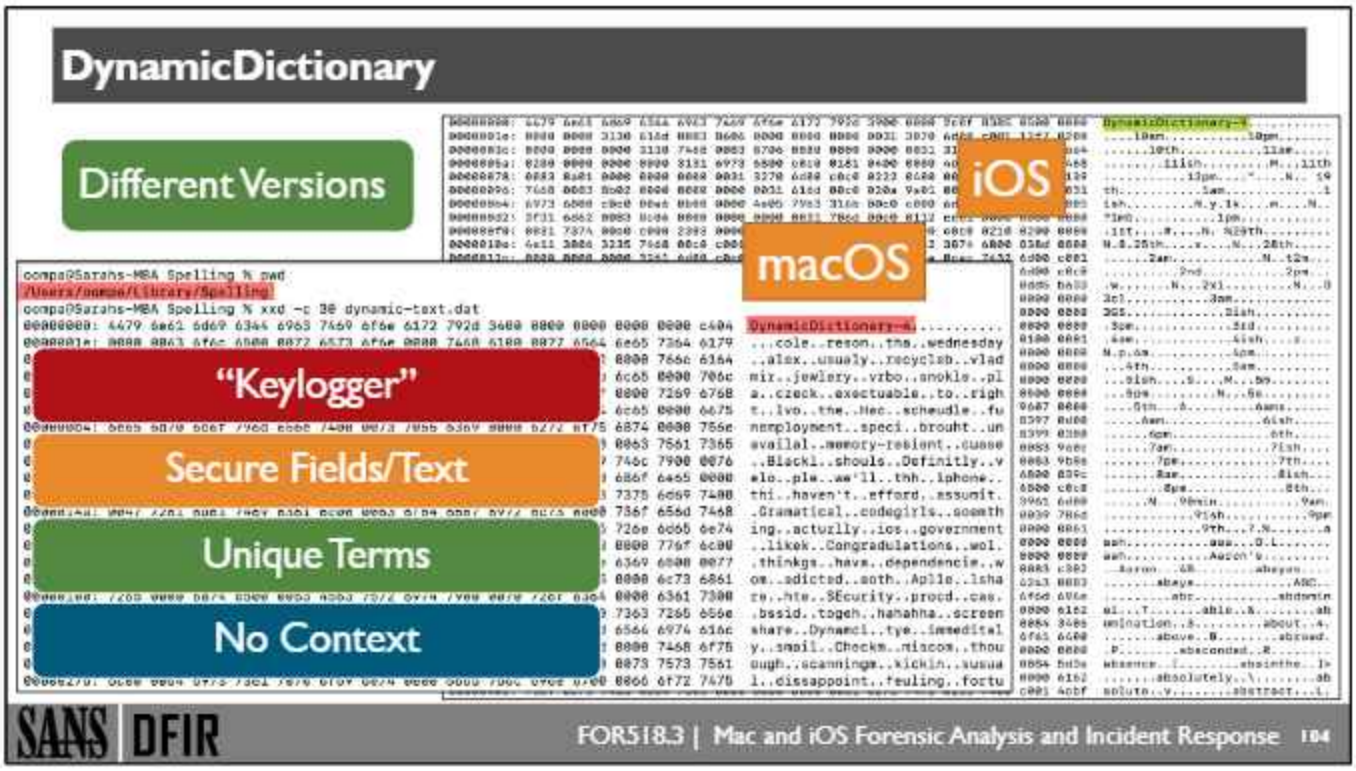
Reference:
http://newosxbook.com/articles/TvOS.html



macOS: ~/Library/Spelling/
iOS:

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

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



macOS: ~/Library/Spelling/
iOS:

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

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

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

iOS SpringBoard

Background Images

- HomeBackgroundThumbnail.jpg
- LockBackgroundThumbnail.jpg

Today View (Widgets)

Dock & Icon State

CarPlay Icon State

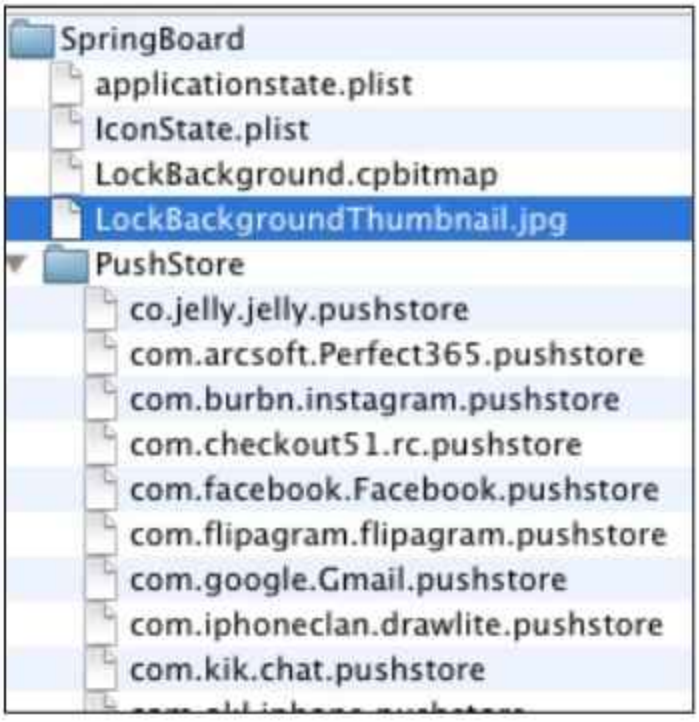
Application Notifications

```
miPhone11:/private/var/mobile/Library/SpringBoard root# ls -1
(null)-CarDisplayDesiredIconState.plist
(null)-CarDisplayIconState.plist
48E58249-F22D-4096-B8D4-1CA81D1DFF1D-CarDisplayIconState.plist
5878D464-8786-4A44-8824-C83F3C013DA6-CarDisplayIconState.plist
58CC10AB-96A1-4CA4-A2E2-34A5A18F9FCF-CarDisplayIconState.plist
707C1ED3-07FC-4AA6-BDFF-624221D32969-CarDisplayIconState.plist
7D2EE4B2-88EC-4D76-9493-C29B6778F7DC-CarDisplayIconState.plist
939BCAAC-7CC3-43E6-9989-7BE357A9939F-CarDisplayIconState.plist
C28F6EB1-751D-4C82-864C-BD846ABC1619-CarDisplayIconState.plist
DownloadingIconImageCache/
EF9B2EC8-A8CE-4BD6-80BC-5696D1E1E4ED-CarDisplayDesiredIconState.plist
EF9B2EC8-A8CE-4BD6-80BC-5696D1E1E4ED-CarDisplayIconState.plist
IconState.plist
LockBackground.cpbitmap
LockBackgroundThumbnail.jpg
LockBackgroundThumbnaildark.jpg
LockoutStateJournal.plist
OriginalLockBackground.cpbitmap
RecentAppLayouts.pb.lzfse
SBLastRestoreIdentifier
TodayViewArchive.plist
```

SANS DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 105

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

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



iOS SpringBoard – Today View and Widgets /private/var/mobile/Library/SpringBoard/TodayViewArchive.plist

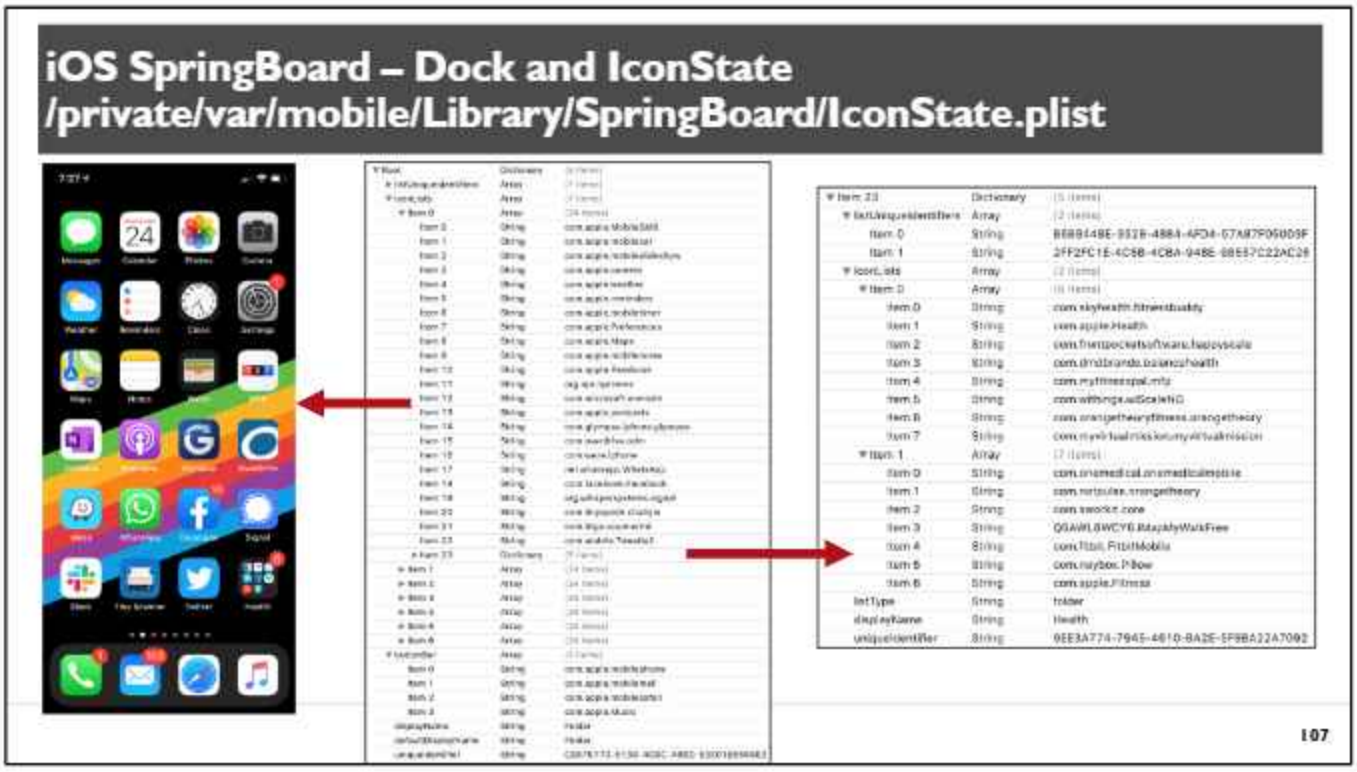
▼ Root	Dictionary	(4 items)
▶ DisplayMode	Dictionary	(4 items)
▶ Known	Array	(99 items)
▶ Penalized	Array	(25 items)
▼ TodayGroup	Array	(8 items)
Item 0	String	com.apple.shortcuts.Today-Extension
Item 1	String	com.waze.iphone.today
Item 2	String	com.apple.Music.RecentlyPlayedTodayExtension
Item 3	String	com.apple.news.widget
Item 4	String	com.apple.Fitness.activity-widget
Item 5	String	com.google.Maps.NearbyTransitTodayExtension
Item 6	String	com.tripit.tripitmobile.todayExtension
Item 7	String	com.apple.ScreenTimeUI.ScreenTimeWidget

Physical: iOS: `/private/var/mobile/Library/SpringBoard/TodayViewArchive.plist`
Backup/FS: iOS: `/mobile/Library/SpringBoard/TodayViewArchive.plist`

The "Today" view was introduced in iOS 12 to keep quick views of different apps and widgets. The configuration of this view is stored in the `TodayViewArchive.plist` file.

Shown above, this view can be user configured to show preferred applications. This organization is stored in the `TodayGroup` key.

Reference:
https://support.apple.com/guide/iphone/add-widgets-to-the-home-screen-iphb8f1bf206/ios



Physical: iOS: `/private/var/mobile/Library/SpringBoard/IconState.plist`
Backup/FS: iOS: `/mobile/Library/SpringBoard/IconState.plist`

SpringBoard controls how each icon and folder is laid out on each screen. The `IconState.plist` file contains an "`Item`" for each screen under "`iconLists`". These are organized from top-to-bottom, left-to-right. In the example above, "`Item 0`" is this device's first home screen. This plist file also shows the contents of the dock in the "`buttonBar`" key, left-to-right.

If a screen contains a folder, there will be additional arrays of application bundle ids, as shown in the screenshot to the right, where a folder named "`Health`" contains many applications in a folder using multiple folder "screens".

iOS SpringBoard – CarPlay Icon State: <GUID>-CarDisplayIconState.plist

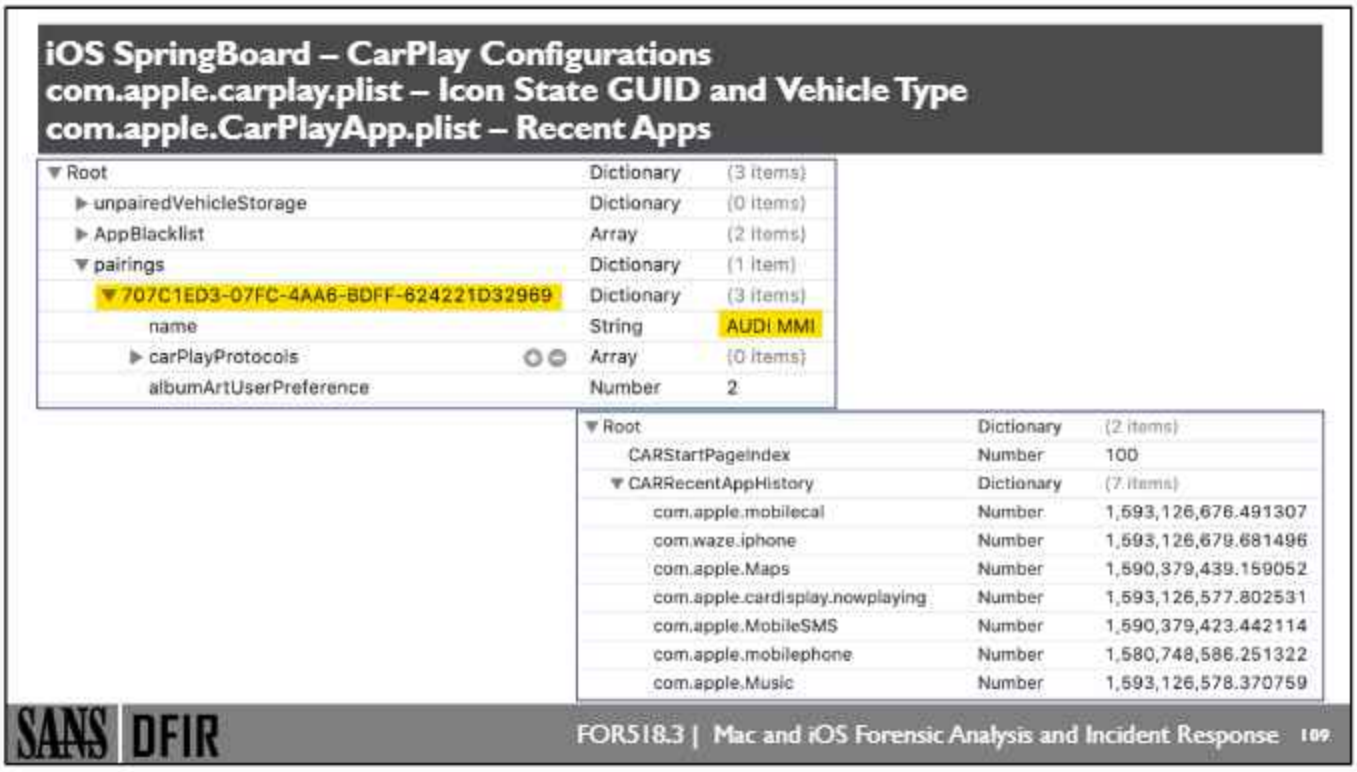
▼ Root	Dictionary	(6 items)
▶ hotUnsupportedIfiers	Array	(3 items)
▼ iconLists	Array	(1 item)
▼ Item 0	Array	(19 items)
Item 0	String	com.apple.mobilephone
Item 1	String	com.apple.Music
Item 2	String	com.apple.Maps
Item 3	String	com.apple.MobileSMS
Item 4	String	us.zoom.videomeetings
Item 5	String	net.whatsapp.WhatsApp
Item 6	String	com.apple.cardisplay.nowplaying
Item 7	String	com.waze.iphone
Item 8	String	com.webex.meeting
Item 9	String	com.apple.cardisplay.OEM
Item 10	String	com.apple.podcasts
Item 11	String	com.apple.iBooks
Item 12	String	com.apple.mobilecal
Item 13	String	com.apple.CarPlaySettings
Item 14	String	com.amazon.mp3.AmazonCloudPlayer
Item 15	String	com.audible.iphone
Item 16	String	com.google.Maps
Item 17	String	com.pandora
Item 18	String	com.spotify.client
▼ metadata	Dictionary	(3 items)
displaysOEMIcon	Boolean	YES
OEMIconLabel	String	Audi MMI
▶ hiddenIcons	Array	(0 items)
displayName	String	Folder
defaultDisplayName	String	Folder
uniqueIdentifier	String	F9A38606-2741-43DF-92F4-7AF74DA0E112

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

CarPlay was introduced in iOS 7. The `<GUID>-CarDisplayIconState.plist` plists will show the organization of these screens. The vehicle description is stored under the metadata key—this was connected to an Audi make of vehicle.

Older versions of this layout tend to persist. The example below shows a previous device of mine connected to a Honda vehicle once, years before, on a road trip!

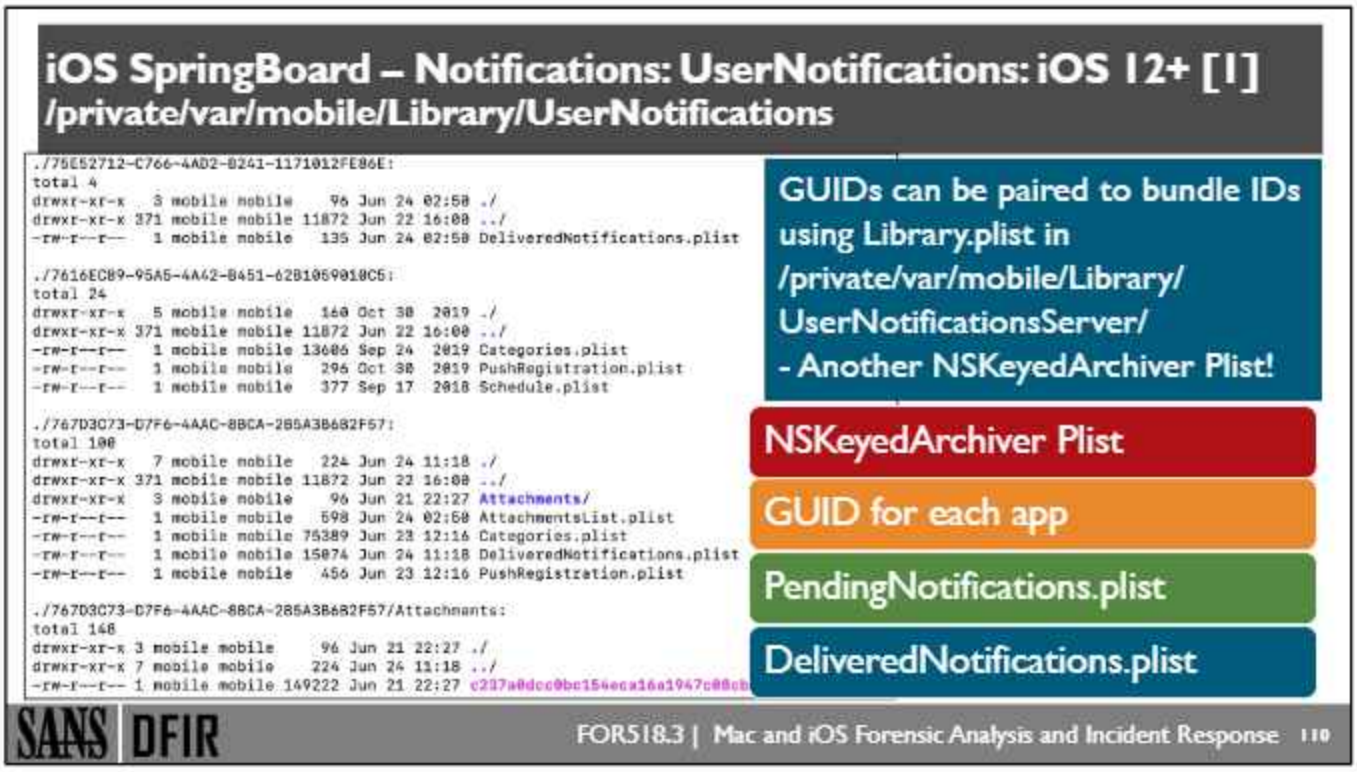
▼ Root	Dictionary	(2 items)
▼ iconLists	Array	(2 items)
▼ Item 0	Array	(8 items)
Item 0	String	com.apple.mobilephone
Item 1	String	com.apple.Music
Item 2	String	com.apple.Maps
Item 3	String	com.apple.MobileSMS
Item 4	String	com.apple.cardisplay.nowplaying
Item 5	String	com.apple.carddisplay.OEM
Item 6	String	com.apple.podcasts
Item 7	String	com.apple.iBooks
▼ Item 1	Array	(7 items)
Item 0	String	com.amazon.mp3.AmazonCloudPlayer
Item 1	String	com.audible.iphone
Item 2	String	com.pandora
Item 3	String	com.spotify.client
Item 4	String	net.whatsapp.WhatsApp
Item 5	String	com.google.Maps
Item 6	String	com.waze.iphone
▼ metadata	Dictionary	(4 items)
OEMIconLabel	String	Honda
maxIconColumnCount	Number	4
▼ hiddenIcons	Array	(0 items)
screenBounds	String	{{0, 0}, {413.33333333333331, 228.66666666666666}}
maxIconRowCount	Number	2
displaysOEMIcon	Boolean	YES



Physical: iOS: `/private/var/mobile/Library/Preferences/`
Backup/FS: iOS: `/mobile/Library/Preferences/`

Since there may be many `CarDisplayIconState.plist` in the SpringBoard directory we need to find the most current one. This can be found in the `com.apple.carplay.plist` file in the /Preferences directory on the device.

The most recently used CarPlay app may also be of interest. Those can be found in the `com.apple.CarPlayApp.plist`.



Physical: `/private/var/mobile/Library/UserNotifications/`
Backup/FS: `/mobile/Library/UserNotifications/`

The notifications that the user may see are stored in a variety of locations depending on the iOS version.

On iOS 12 and newer they are stored in the `UserNotifications` directory. Under this directory will be a GUID for each app that may have notifications with a variety of plists under it. The `PendingNotifications.plist` and `DeliveredNotifications.plist` files are of interest to us.

In these directories, you may find a few plists for each app. The `DeliveredNotifications.plist` contains the notifications that we might be looking for. Some of these may have associated attachments (usually pictures) that may be stored in an `Attachments` directory.

To pair the GUIDs with their associated app bundle ID, you can look at the Library.plist in /private/var/mobile/Library/UserNotificationsServer/. This file does not appear to be in iOS backups; however, looking into some of the plists, you can sometimes find hints of what application it is associated with.

iOS SpringBoard – Notifications: UserNotifications: iOS 12+ [2]

```
45 => "ShouldUseRequestIdentifierForDismissalSync"
46 => "ShouldHideTime"
47 => "Isolated severe storms later today could produce hail, damaging winds. Check timing, local impacts in the app."
48 => 0
49 => 0
50 => 1
51 => 1
52 => "90462921-586D-413C-9105-ECF723023E29"
53 => {
```

DeliveredNotifications.plist

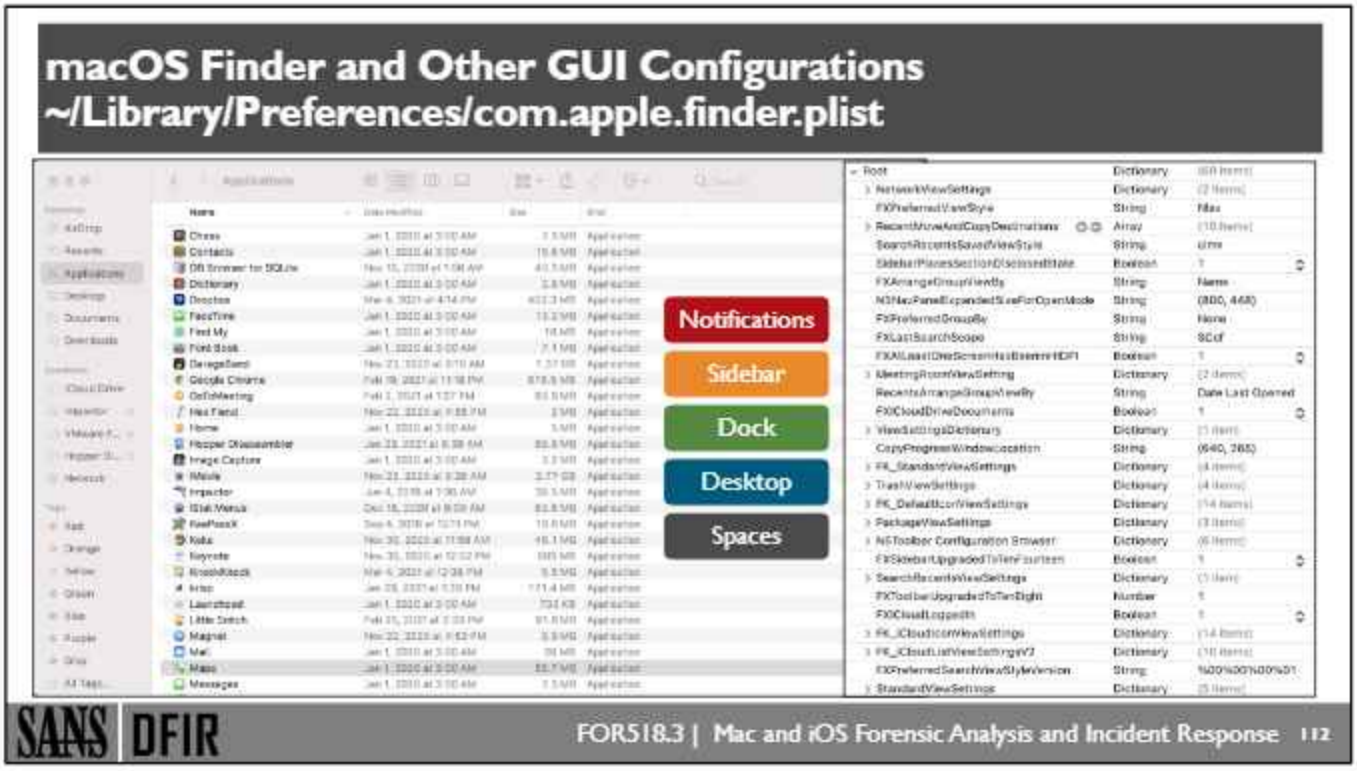
```
44 => "AppNotificationSummaryArgument"
45 => "AppNotificationTitle"
46 => "ShouldUseRequestIdentifierForDismissalSync"
47 => "ShouldHideTime"
48 => "You received $0.45 in Daily Cash yesterday and your June total is $17.30."
49 => 0
50 => 0
51 => 1
52 => {
  "$class" => <CFKeyedArchiverUID 0x7f93dce078e0 [0x7fff8b5e68c0]>(value = 53)
  "NS.time" => 614174585
}
53 => {
  "$classes" => [
    0 => "NSDate"
    1 => "NSObject"
  ]
  "$classname" => "NSDate"
}
```


Physical: `/private/var/mobile/Library/UserNotifications/`
Backup/FS: `/mobile/Library/UserNotifications/`

The notification plist files are stored as NSKeyedArchiver plist files. A couple of examples are shown above.

- One notification for Accuweather
- One notification for Apple Cash | Wallet

If the notifications are cleared by the user, they are removed from the plist.



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

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

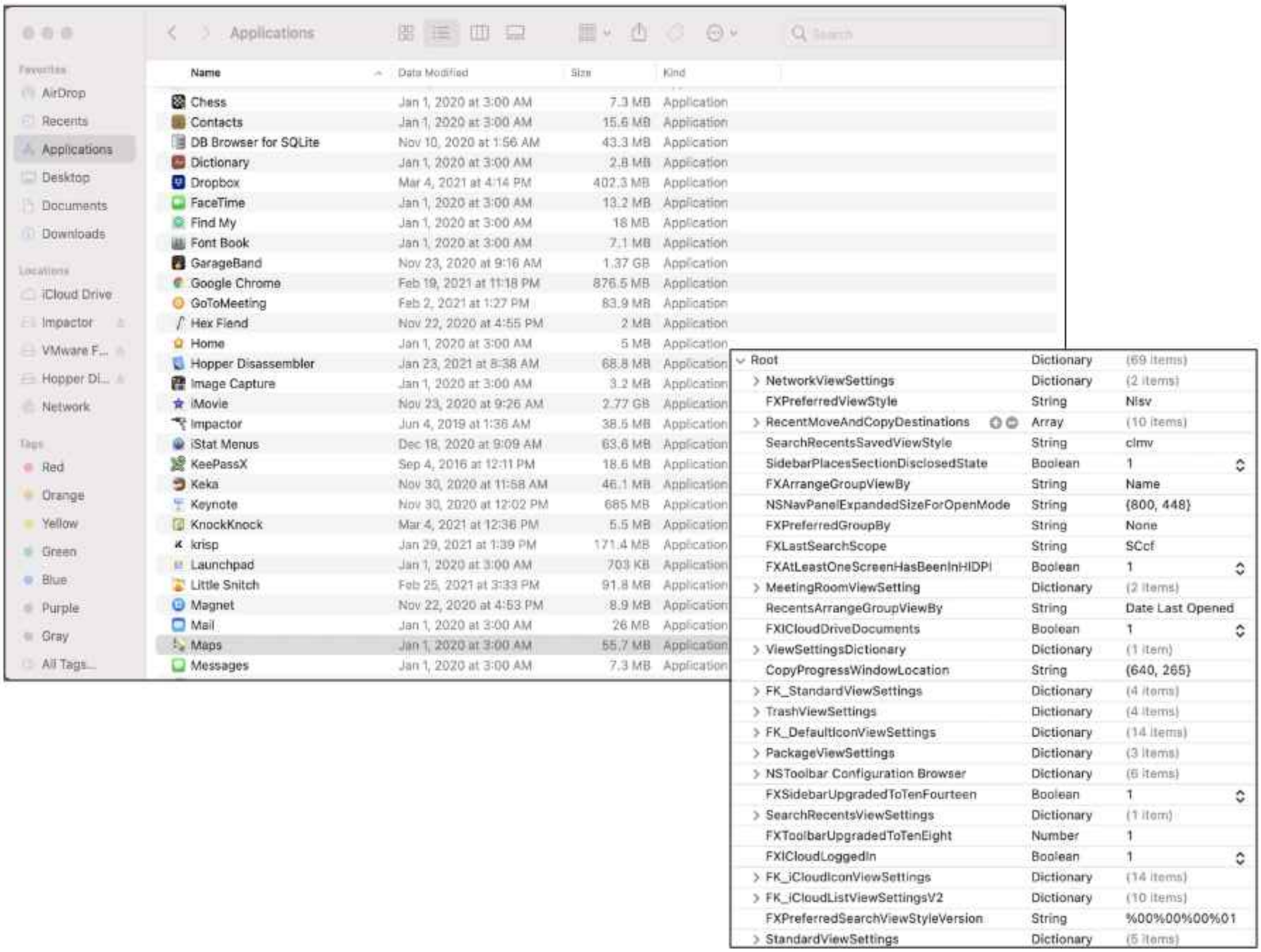
- Favorites: By default, the user directories are shown, such as Documents, Pictures, Music, Downloads, etc.
- Locations: Contains the volumes the system has access to: The host volume (Macintosh HD), USB drives, mounted DMGs, etc.

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

For example:

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

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



macOS Notifications – NotificationCenter
/private/var/folders/<DARWIN_USER_DIR>/com.apple.notificationcenter/db2/db

```
root@Sarahs-MBA com.apple.notificationcenter # pwd
/System/Volumes/Data/private/var/folders/8j/5m02j58n1qq4k9q0lh6sd4rh0000gn/0/com.apple.notificationcenter
root@Sarahs-MBA com.apple.notificationcenter # ls -laR
total 0
drwx------   4 oompa  staff  128 Oct 30  2019 .
drwxr-xr-x@ 19 oompa  staff  608 Apr 23 09:40 ..
drwx------   3 oompa  staff   96 Dec 29 17:31 attachments
drwx------   5 oompa  staff  160 Oct 30  2019 db2

./attachments:
total 7672
drwx------  3 oompa  staff       96 Dec 29 17:31 .
drwx------  4 oompa  staff      128 Oct 30  2019 ..
-rw-r--r--@ 1 oompa  staff  3924621 Dec 29 09:06 ec42019b85e9c99f0947148 6caab5267972defc4.jpeg

./db2:
total 6088
drwx------  5 oompa  staff      160 Oct 30  2019 .
drwx------  4 oompa  staff      128 Oct 30  2019 ..
-rw-r--r--  1 oompa  staff  1372160 Jun 26 15:52 db
-rw-r--r--  1 oompa  staff    32768 Jun 16 16:49 db-shm
-rw-r--r--  1 oompa  staff   894872 Jun 26 18:22 db-wal
```


The notification center database on macOS is in the user's DARWIN_USER_DIR path. This path will be different for each user on the system and is generated using a specific algorithm (please see the referenced article).

The database is simply named 'db' and is in the `com.apple.notificationcenter/db2/` directory structure. If notifications have attachments, you may find those in the `/attachments` directory.

Reference:
https://www.swiftforensics.com/2017/04/the-mystery-of-varfolders-on-osx.html

macOS Notifications – NotificationCenter Database – 'db'

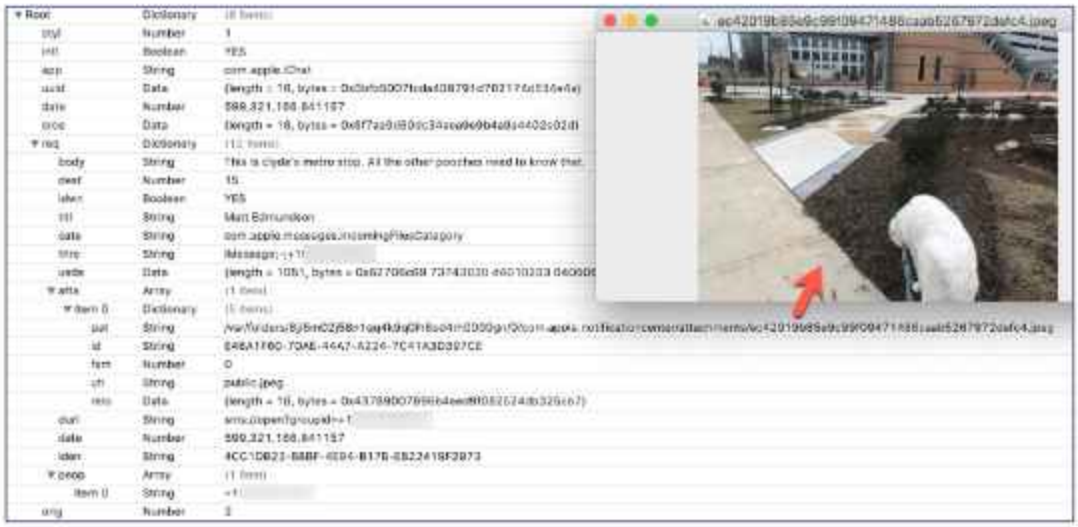
```
SELECT
    DATETIME(RECORD.DELIVERED_DATE+978307200,'UNIXEPOCH') AS 'DATE DELIVERED',
    APP.IDENTIFIER AS 'BUNDLE ID',
    APP.BADGE AS 'APP BADGE',
    RECORD.PRESENTED AS 'PRESENTED',
    RECORD.STYLE AS 'STYLE',
    RECORD.SNOOZE_FIRE_DATE AS 'SNOOZE FIRE DATE',
    RECORD.DATA AS 'NOTIFICATION DATA',
    CATEGORIES.CATEGORIES AS 'CATEGORIES',
    RECORD.REQUEST_DATE AS 'REQUEST DATE',
    RECORD.REQUEST_LAST_DATE AS 'REQUEST LAST DATE',
    HEX(RECORD.UUID) AS 'UUID',
    RECORD.REC_ID AS "RECORD TABLE ID"
FROM RECORD
LEFT JOIN APP ON APP.APP_ID == RECORD.APP_ID
LEFT JOIN CATEGORIES ON CATEGORIES.APP_ID == RECORD.APP_ID
```

	DATE DELIVERED	BUNDLE ID	APP BADGE	PRESENTED	STYLE	SNOOZE FIRE DATE	NOTIFICATION DATA	CATEGORIES	REQUEST DATE	REQUEST LAST DATE
36	2019-12-23 22:41:48	com.apple.photos	NULL	0	1	NULL	BLOB	BLOB	NULL	NULL
37	2019-12-23 22:43:37	com.apple.photos	NULL	0	1	NULL	BLOB	BLOB	NULL	NULL
38	2019-12-23 22:50:54	com.apple.photos	NULL	0	1	NULL	BLOB	BLOB	NULL	NULL
39	2018-12-21 17:32:15	at.obdev.littlesnitchnetworkmonitor	NULL	1	1	NULL	BLOB	NULL	NULL	NULL
40	2018-12-21 17:32:24	at.obdev.littlesnitchnetworkmonitor	NULL	1	1	NULL	BLOB	NULL	NULL	NULL
41	2019-12-21 18:09:38	at.obdev.littlesnitchnetworkmonitor	NULL	1	1	NULL	BLOB	NULL	NULL	NULL
42	2019-12-29 14:06:06	com.apple.ichat	0	0	1	NULL	BLOB	BLOB	NULL	NULL
43	2019-12-29 22:43:51	at.obdev.littlesnitchnetworkmonitor	NULL	1	1	NULL	BLOB	NULL	NULL	NULL
44	2020-01-06 15:57:58	at.obdev.littlesnitchnetworkmonitor	NULL	1	1	NULL	BLOB	NULL	NULL	NULL

The database keeps track of the notification delivery date, app bundle IDs, presentation, and style. The notification on row 42 was delivered on 12/29/2019 for the Messages (com.apple.ichat) application.

The BLOB data in "NOTIFICATION DATA" contains an embedded binary plist with contextual information about this notification.

macOS Notifications – NotificationCenter Embedded Notification Plist



The embedded plist was extracted and saved to show the contents of it. We can see useful items such as the text of the message, timestamps, whom it was from, their contact information (blurred), and attachment path data.

This attachment was still on the device ~7 months later!

Saved Application State

Introduced in 10.7 as "resume" feature

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

Locations:

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

```
compa@MBP-M1 Saved Application State % pwd
/Users/compa/Library/Saved Application State
compa@MBP-M1 Saved Application State % ls -l
total 0
drwx------  4 compa  staff  128 Aug 30 09:32 com.Google.GoogleEarthPro.savedState
drwx------  6 compa  staff  192 Nov 21 17:06 com.TechSmith.Snagit2021.savedState
lrwxr-xr-x  1 compa  staff  117 Jun 23 08:06 com.apple.AppStore.savedState -> /Users/compa/Library/Containers/com.apple.AppStore/Data/Library/Saved Application State/com.apple.AppStore.savedState
drwx------  4 compa  staff  128 Nov 19 02:42 com.apple.InstallAssistant.macOSMonterey.savedState
lrwxr-xr-x  1 compa  staff  169 Jun 16 15:39 com.apple.Maps.savedState -> /Users/compa/Library/Containers/com.apple.Maps/Data/Library/Saved Application State/com.apple.Maps.savedState
lrwxr-xr-x  1 compa  staff  119 Nov 27  2020 com.apple.MobileSMS.savedState -> /Users/compa/Library/Containers/com.apple.MobileSMS/Data/Library/Saved Application State/com.apple.MobileSMS.savedState
lrwxr-xr-x  1 compa  staff  111 Nov 29  2020 com.apple.Notes.savedState -> /Users/compa/Library/Containers/com.apple.Notes/Data/Library/Saved Application State/com.apple.Notes.savedState
lrwxr-xr-x  1 compa  staff  113 May  9  2021 com.apple.Photos.savedState -> /Users/compa/Library/Containers/com.apple.Photos/Data/Library/Saved Application State/com.apple.Photos.savedState
lrwxr-xr-x  1 compa  staff  115 Nov 29  2020 com.apple.Preview.savedState -> /Users/compa/Library/Containers/com.apple.Preview/Data/Library/Saved Application State/com.apple.Preview.savedState
lrwxr-xr-x  1 compa  staff  123 Apr 26  2021 com.apple.QuickTimePlayerX.savedState -> /Users/compa/Library/Containers/com.apple.QuickTimePlayerX/Data/Library/Saved Application State/com.apple.QuickTimePlayerX.savedState
lrwxr-xr-x  1 compa  staff  113 Nov 21  2020 com.apple.Safari.savedState -> /Users/compa/Library/Containers/com.apple.Safari/Data/Library/Saved Application State/com.apple.Safari.savedState
drwx------  4 compa  staff  128 Nov 19 02:42 com.apple.ScriptEditor2.savedState
drwx------  8 compa  staff  192 Nov 22 10:16 com.apple.Terminal.savedState
```

SANS DFIR | FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 117

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

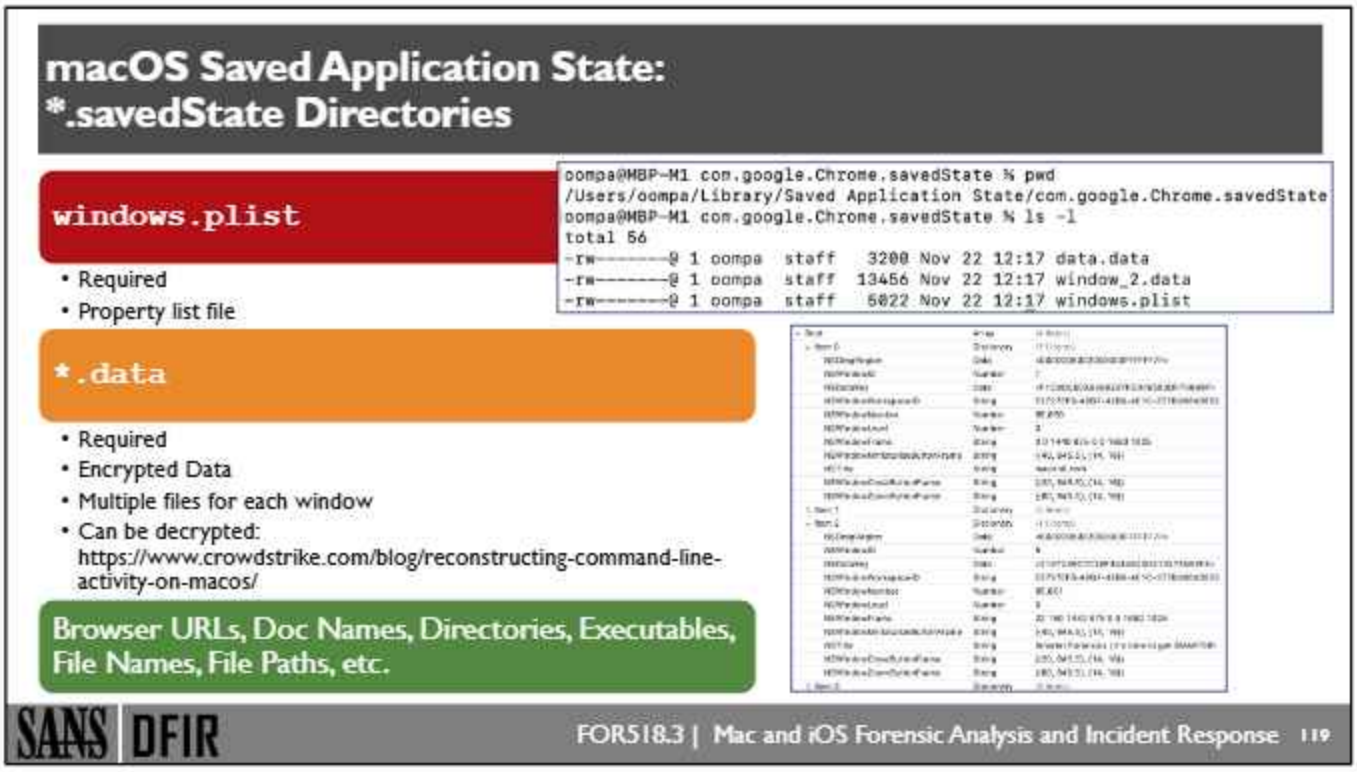
The directory `~/Library/Saved Application State/` also contains links to the `.savedState` directories in the sandboxed application containers.

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

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

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

```
oompa@MBP-M1 Saved Application State % pwd
/Users/oompa/Library/Saved Application State
oompa@MBP-M1 Saved Application State % ls -l
total 0
drwx------  4 oompa  staff  128 Aug 30 09:32 com.Google.GoogleEarthPro.savedState
drwx------  6 oompa  staff  192 Nov 21 17:08 com.TechSmith.Snagit2021.savedState
lrwxr-xr-x  1 oompa  staff  117 Jun 23 08:06 com.apple.AppStore.savedState -> /Users/oompa/Library/Containers/com.apple.A
ppStore/Data/Library/Saved Application State/com.apple.AppStore.savedState
drwx------  4 oompa  staff  128 Nov 19 02:42 com.apple.InstallAssistant.macOSMonterey.savedState
lrwxr-xr-x  1 oompa  staff  109 Jun 18 15:39 com.apple.Maps.savedState -> /Users/oompa/Library/Containers/com.apple.Maps/
Data/Library/Saved Application State/com.apple.Maps.savedState
lrwxr-xr-x  1 oompa  staff  119 Nov 22  2020 com.apple.MobileSMS.savedState -> /Users/oompa/Library/Containers/com.apple.
MobileSMS/Data/Library/Saved Application State/com.apple.MobileSMS.savedState
lrwxr-xr-x  1 oompa  staff  111 Nov 29  2020 com.apple.Notes.savedState -> /Users/oompa/Library/Containers/com.apple.Note
s/Data/Library/Saved Application State/com.apple.Notes.savedState
lrwxr-xr-x  1 oompa  staff  113 May  9  2021 com.apple.Photos.savedState -> /Users/oompa/Library/Containers/com.apple.Pho
tos/Data/Library/Saved Application State/com.apple.Photos.savedState
lrwxr-xr-x  1 oompa  staff  115 Nov 29  2020 com.apple.Preview.savedState -> /Users/oompa/Library/Containers/com.apple.Pr
eview/Data/Library/Saved Application State/com.apple.Preview.savedState
lrwxr-xr-x  1 oompa  staff  133 Apr 20  2021 com.apple.QuickTimePlayerX.savedState -> /Users/oompa/Library/Containers/com
.apple.QuickTimePlayerX/Data/Library/Saved Application State/com.apple.QuickTimePlayerX.savedState
lrwxr-xr-x  1 oompa  staff  113 Nov 22  2020 com.apple.Safari.savedState -> /Users/oompa/Library/Containers/com.apple.Saf
ari/Data/Library/Saved Application State/com.apple.Safari.savedState
drwx------  4 oompa  staff  128 Nov 19 02:42 com.apple.ScriptEditor2.savedState
drwx------  6 oompa  staff  192 Nov 22 10:16 com.apple.Terminal.savedState
```



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

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

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

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

These encrypted *.data files can be decrypted – a great use for using these in an investigation was done by CrowdStrike - https://www.crowdstrike.com/blog/reconstructing-command-line-activity-on-macos/.

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

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

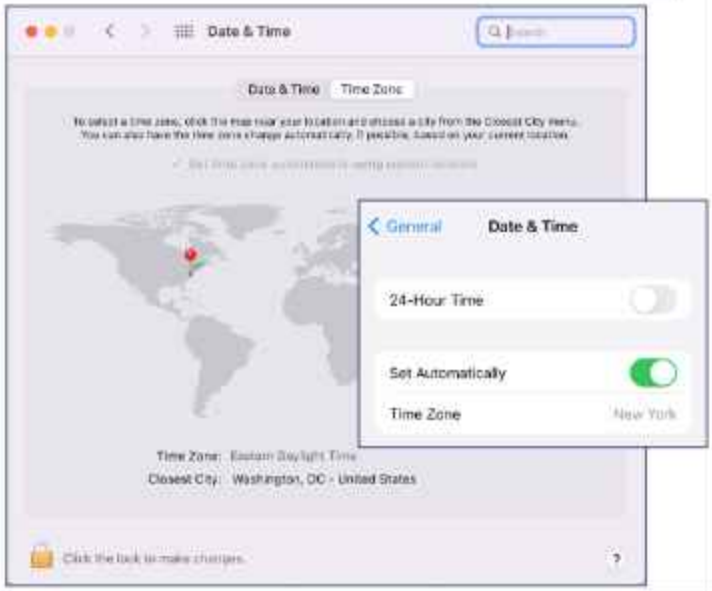
Lots of good forensic tidbits can be found looking at the `windows.plist` file for an application.

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

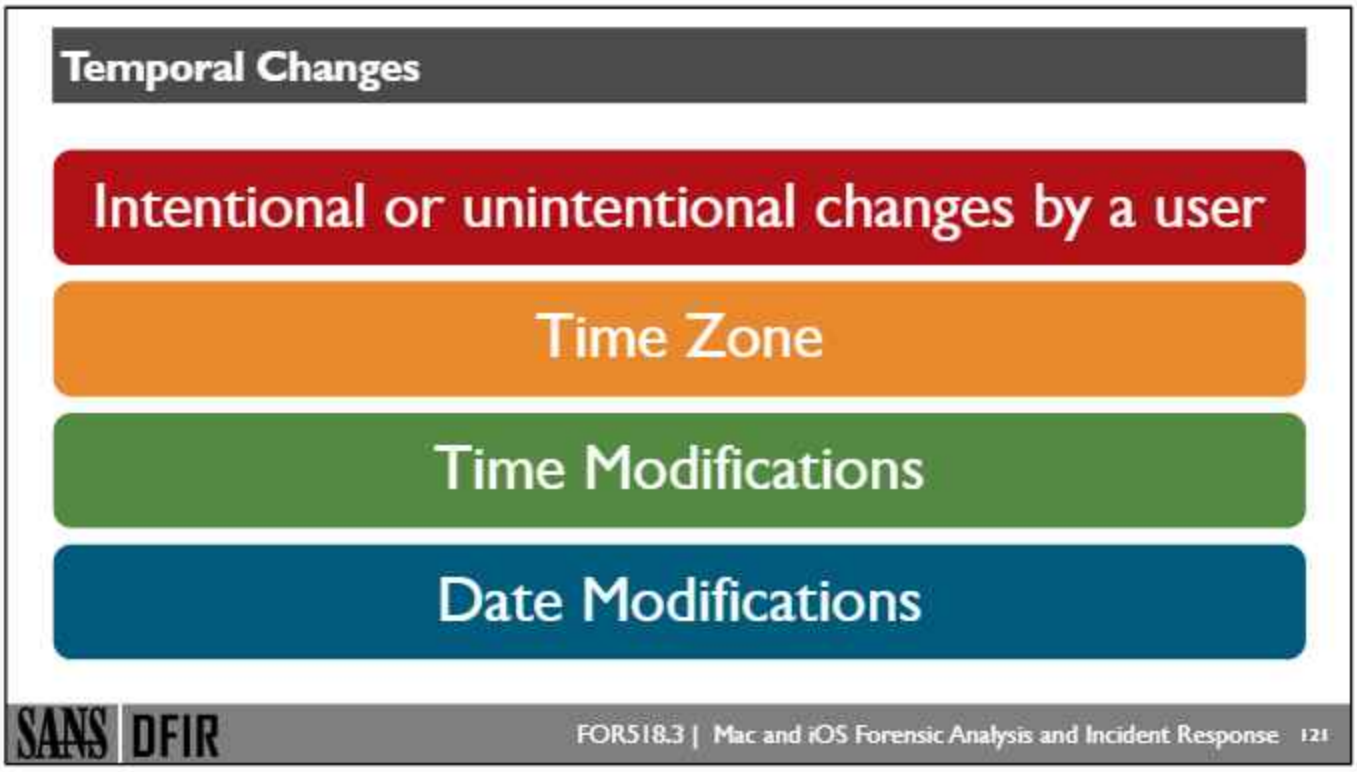
Many `window.plist` files have a key named `NSExecutableInode`. This is used to store the iNode number of the related application executable.

Date & Time

- Recall:
  - /etc/localtime
  - .GlobalPreferences.plist
  - com.apple.timezone.auto.plist
  - iOS general.log (header)



The Date & Time Preference Panel can be used to configure date and time preferences and time zones.



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

Time Changes: Location-Based system.log and Unified

- Location services
- Network-based lookup
- Travelling, changes upon first network connection
- Setting in system preferences: com.apple.timezone.auto.plist
- Non-location-based indicators
  - Timestamp jumps in /var/log/*
  - /etc/localtime symlink timestamps

```
2017-02-05 15:31:06.644564-0500  localhost locationd[87]: (locationd) Created Activity ID: 0x800000000d9d98, Description: SecTrustEvaluateIfNecessary
2017-02-05 15:31:06.647918-0500  localhost locationd[87]: (locationd) Created Activity ID: 0x800000000d9d99, Description: SecTrustEvaluateIfNecessary
2017-02-05 12:31:06.641456-0800  localhost locationd[87]: [com.apple.locationd.Position.WifiPosition] TlurAssoc, Request, <private>, urgent, <private>, type, <private>
2017-02-05 12:31:06.641470-0800  localhost locationd[87]: [com.apple.locationd.Position.WifiPosition] TlurState, StartScan, <private>, state, <private>
2017-02-05 12:31:06.642373-0800  localhost timezoned[9463]: (CoreLocation) [com.apple.locationd.Core.Client] New location is identical to old location, discarding
2017-02-05 15:31:06.757196-0500  localhost locationd[87]: Location icon should now be in state 'Active'
2017-02-05 12:31:07.218975-0800  localhost locationd[87]: [com.apple.locationd.Position.WifiPosition] TlurState, ScanNotify, <private>, aps, <private>, state, <private>
2017-02-05 15:31:07.242883-0500  localhost locationd[87]: (locationd) Created Activity ID: 0x800000000d9d9a, Description: Loading Preferences From System CFPrefsD For Search List
```

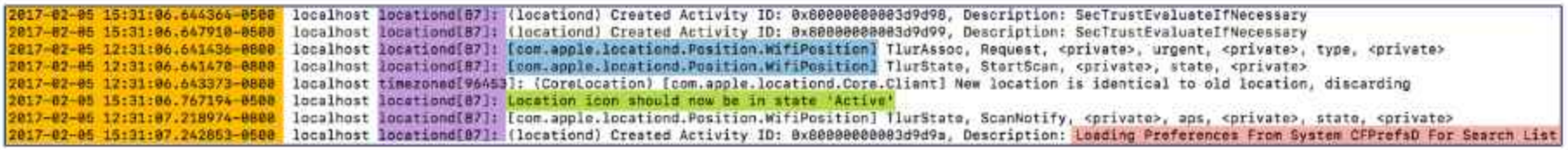

If location-based services are used, the macOS system will perform a lookup of where the device is (assuming it is connected to the network) in an attempt to determine its location.

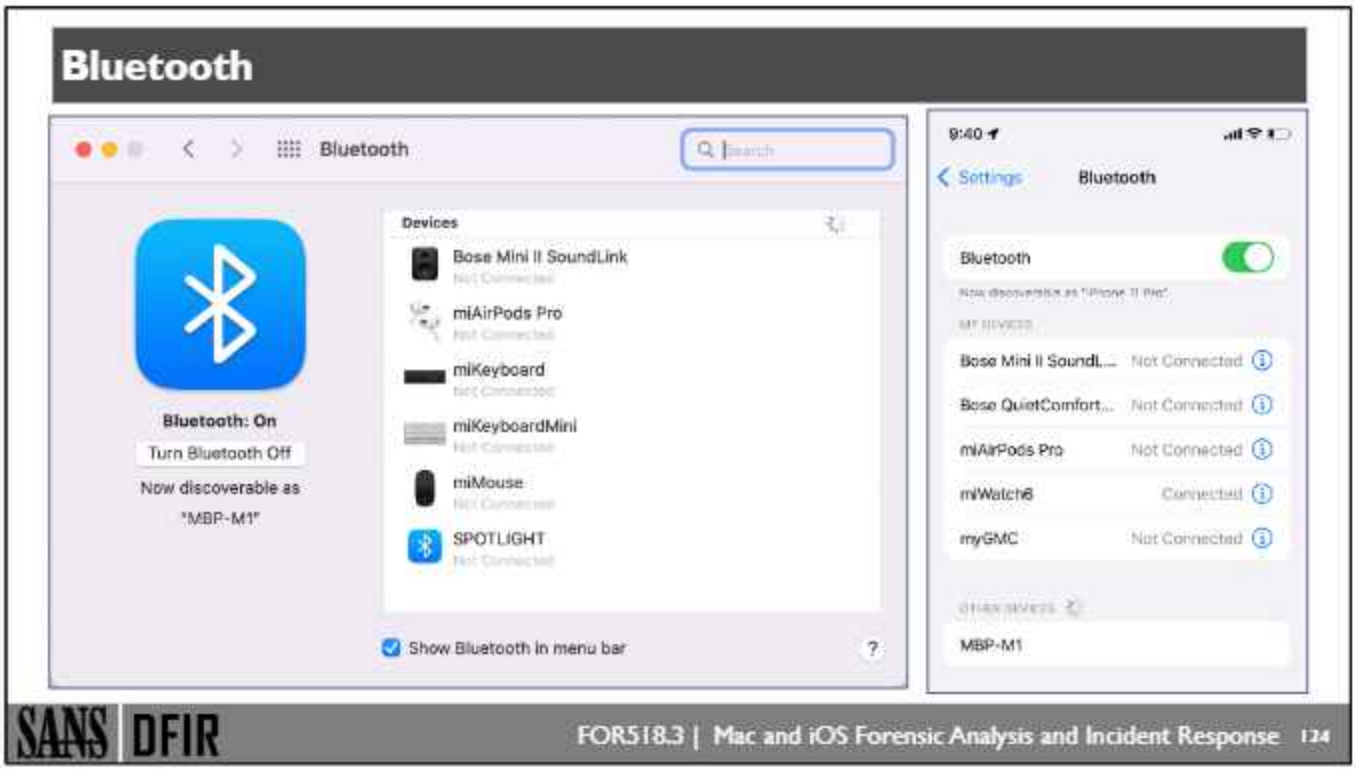
In the logs, search for "`location`" (the location services daemon) or "`timezoned`" (the time zone daemon). Other indicators in the logs may suggest the time is being changed, such as the logging timestamp and other messages.

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

You can also see timestamp "jumps" in the logs located in /var/log because those timestamps are stored in local system time.

The /etc/`localtime` symlink is updated when the time zone is changed to reflect the new time zone.

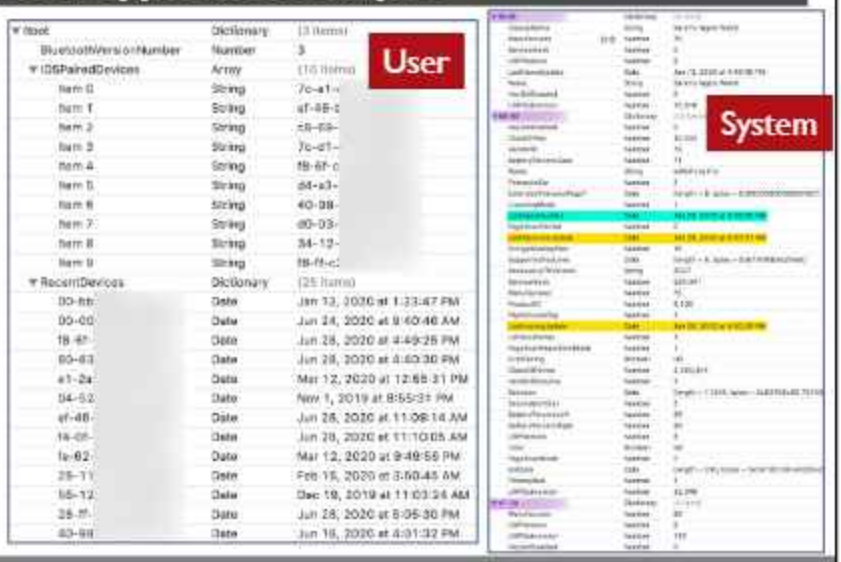




The Bluetooth Preference Panel can be used to configure Bluetooth devices.

macOS Bluetooth Devices (macOS 11-)
User: ~/Library/Preferences/ByHost/com.apple.Bluetooth.<HWUUID>.plist
System: /Library/Preferences/com.apple.Bluetooth.plist

- Organized by Bluetooth MAC Address
- May have limited data
- 'Last' Used Time:
  - User: 'RecentDevices'
  - System: 'LastQueryUpdate', 'LastServicesUpdate'
- "First" Used Time:
  - System: 'LastNameUpdate'
- Beware Timestamps!
  - Use logs (bluetoothd)



The system's `com.apple.Bluetooth.plist` property list located in the `/Library/Preferences/` directory contains the Bluetooth device cache in the `DeviceCache` key and the paired devices in the `PairedDevices` key. This `DeviceCache` key contains a history of the Bluetooth devices connected to the system, however, devices can easily be removed.

The user's `com.apple.Bluetooth.<HW_UUID>.plist` contains devices that a user specifically connected to.

Using these two plist files, we can correlate and find devices of interest. Note that some Bluetooth devices, especially those in an Apple ecosystem environments, may be associated without the user specifically setting up the device in the Bluetooth Preference pane up due to Continuity associated devices.

Timestamps should be used carefully; certain user actions can change how these times may be interpreted. For example, if I changed the name of my AirPods, the timestamp generally used as a "first" connection time will be updated. These may not necessarily be reliable for the specific action you are attempting to investigate. The "Last" used time is when the Bluetooth services have been updated. A user may still be listening on those AirPods for hours after.

User

▼ Root	Dictionary	(3 items)
BluetoothVersionNumber	Number	3
▼ IDSPairedDevices	Array	(10 items)
Item 0	String	7c-a1-
Item 1	String	ef-46-b
Item 2	String	c8-69-
Item 3	String	7c-d1-
Item 4	String	f8-6f-c
Item 5	String	d4-a3-
Item 6	String	40-98-
Item 7	String	d0-03-
Item 8	String	34-12-
Item 9	String	f8-ff-c2
▼ RecentDevices	Dictionary	(25 items)
00-bb	Date	Jan 13, 2020 at 1:23:47 PM
00-00	Date	Jun 24, 2020 at 9:40:46 AM
f8-6f-	Date	Jun 28, 2020 at 4:49:25 PM
60-83	Date	Jun 28, 2020 at 4:40:30 PM
e1-2a	Date	Mar 12, 2020 at 12:55:31 PM
04-52	Date	Nov 1, 2019 at 8:55:31 PM
ef-46-	Date	Jun 28, 2020 at 11:09:14 AM
f4-0f-	Date	Jun 28, 2020 at 11:10:05 AM
fe-62-	Date	Mar 12, 2020 at 9:49:56 PM
28-11	Date	Feb 15, 2020 at 3:50:45 AM
56-12	Date	Dec 19, 2019 at 11:03:24 AM
28-ff-	Date	Jun 28, 2020 at 5:05:30 PM
40-98	Date	Jun 18, 2020 at 4:01:32 PM

System

▼ f8-6f-	Dictionary	(8 items)
displayName	String	Sarah's Apple Watch
Manufacturer	Number	76
ServiceMask	Number	0
LMPVersion	Number	9
LastNameUpdate	Date	Jun 16, 2020 at 4:49:40 PM
Name	String	Sarah's Apple Watch
HeySiriEnabled	Number	0
LMPSubversion	Number	15,616
▼ 60-83	Dictionary	(34 items)
HeySiriEnabled	Number	0
ClockOffset	Number	30,304
VendorID	Number	76
BatteryPercentCase	Number	71
Name	String	miAirPods Pro
PrimaryInEar	Number	2
ExtendedFeaturesPage1	Data	(length = 8, bytes = 0x0000000000000007)
ListeningMode	Number	1
LastNameUpdate	Date	Jun 28, 2020 at 4:02:05 PM
PageScanPeriod	Number	0
LastServicesUpdate	Date	Jun 28, 2020 at 4:40:31 PM
EncryptionKeySize	Number	16
SupportedFeatures	Data	(length = 8, bytes = 0x877bffdbfe2ffebf)
AccessoryFWVersion	String	2D27
ServiceMask	Number	524,441
Manufacturer	Number	76
ProductID	Number	8,206
RightDoubleTap	Number	3
LastInquiryUpdate	Date	Jun 28, 2020 at 4:40:29 PM
LeftDoubleTap	Number	3
PageScanRepetitionMode	Number	1
FirstPairing	Boolean	NO
ClassOfDevice	Number	2,360,344
VendorIDSource	Number	1
Services	Data	(length = 11240, bytes = 0x62706c69 73743
SecondaryInEar	Number	2
BatteryPercentLeft	Number	99
BatteryPercentRight	Number	99
LMPVersion	Number	9
InEar	Boolean	NO
PageScanMode	Number	0
EIRData	Data	(length = 240, bytes = 0x09100100 4c000e2
PrimaryBud	Number	2
LMPSubversion	Number	32,049
▼ e1-2a	Dictionary	(4 items)
Manufacturer	Number	89
LMPVersion	Number	8
LMPSubversion	Number	135
HeySiriEnabled	Number	0

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

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

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

iOS Bluetooth Devices and macOS 12

iOS: /private/var/containers/Shared/SystemGroup/<GUID>
macOS 12: /Library/Bluetooth/

- /Library/Preferences/com.apple.MobileBluetooth.devices.plist
- [/Library/Databases/]com.apple.MobileBluetooth.ledevices.other.db
- [/library/Databases/]com.apple.MobileBluetooth.ledevices.paired.db

Paired Devices

- Cars
- iDevices
- Computers
- Speakers/Headphones
- Bluetooth VS. Bluetooth LE

Timestamp

- "LastSeen" - Last Used
- Warning! Timestamp stored in "local time" Unix Epoch

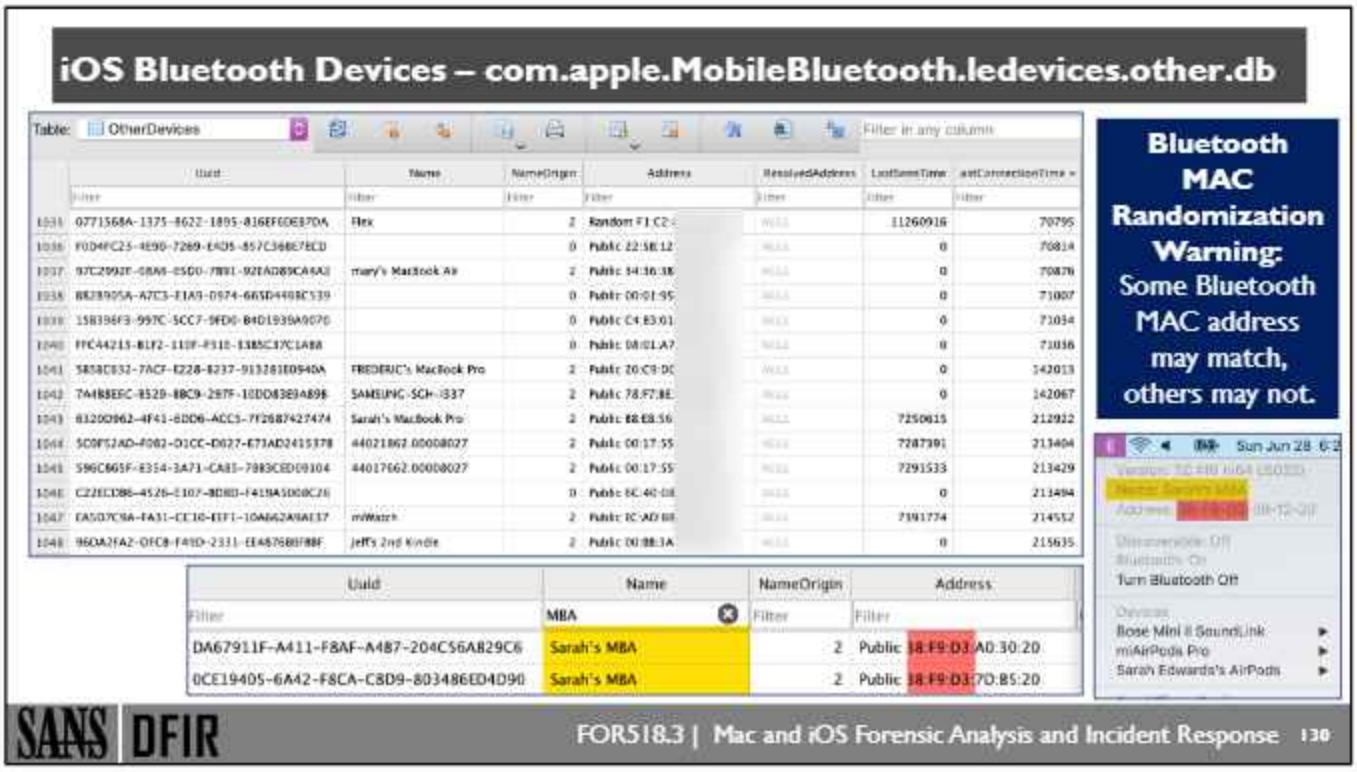
On iOS devices and now with macOS 12, there is a plist, com.apple.MobileBluetooth.devices.plist keeping track of connected Bluetooth devices as well as two SQLite databases for Bluetooth Low Energy devices keeping track of Bluetooth Low Energy devices.

- com.apple.MobileBluetooth.ledevices.other.db
- com.apple.MobileBluetooth.ledevices.paired.db

You may find all types of devices in these files, some more interesting than others but a good place to find other areas of investigative value.

In the screenshot above is an example from com.apple.MobileBluetooth.devices.plist of where this device connected to a GMC vehicle. The Unix Epoch timestamp in "LastSeen" is when this device was last used. This timestamp is different than most as it is storing its time in local system time.

▼ 98:D3	Dictionary	(0 items)
▼ A8:BE:	Dictionary	(0 items)
▼ F8:16:	Dictionary	(0 items)
▼ 00:15:	Dictionary	(0 items)
▼ 90:F1:	Dictionary	(0 items)
▼ B8:9F:09:D3:2D:EF	Dictionary	(28 items)
DefaultName	String	Handsfree
LastHandsfreeVersion	Data	{length = 2, bytes = 0x0601}
ServiceA2DP	String	Unknown
PhonebookSyncSettings	Number	31
LastAVRCPControllerSupportedFea...	Data	{length = 2, bytes = 0xc103}
ServiceAACP	String	Unknown
ServiceRemote	String	Unknown
ServiceNetSharingUser	String	Unknown
ServiceBraille	String	Unsupported
LastSeenTime	Number	1,584,627,905
Name	String	myGMC
DeviceClass	Data	{length = 4, bytes = 0x08043400}
ServiceWiAP	String	Unknown
ServiceWirelessCarPlay	String	Unknown
ServicePhoneBook	String	Unsupported
EncryptionKeySize	Data	{length = 1, bytes = 0x10}
LastAVRCPControllerVersion	Data	{length = 2, bytes = 0x0601}
LastAVRCPVersion	Data	{length = 2, bytes = 0x0601}
PhonebookSyncGroup	Number	-1
LastHandsfreeSupportedFeatures	Data	{length = 2, bytes = 0x3f00}
ServiceGATT	String	Unknown
initiateSDPMirroringState	Boolean	NO
ServiceHandsfree	String	Supported
ServiceHID	String	Unsupported
ServiceMAP	String	Supported
LastAVRCPTargetVersion	Data	{length = 2, bytes = 0x0601}
LastAVRCPTargetSupportedFeatures	Data	{length = 2, bytes = 0x0200}
ServiceGaming	String	Unsupported



The com.apple.MobileBluetooth.ledevices.other.db database contains Bluetooth LE devices that are "seen" but not necessarily paired with. My own database has devices that I've used before on the iPhone that this example came from.

Some Bluetooth MAC addresses may be the actual MAC address or possibly a randomized one as shown above with "Sarah's MBA".

While it doesn't have every Bluetooth device ever seen, it could be a potential place to find devices in the nearby vicinity.

iOS Bluetooth Devices – com.apple.MobileBluetooth.ledevices.paired.db

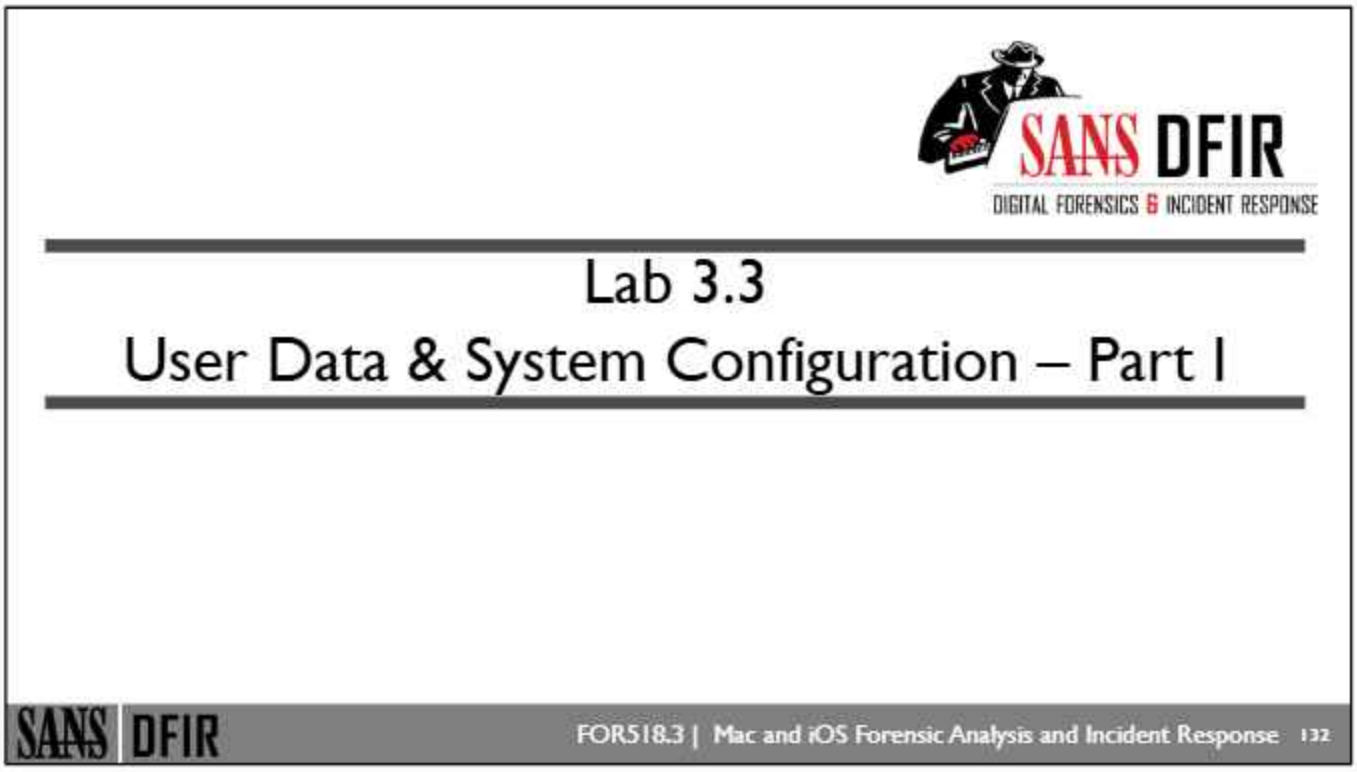
May not include ALL paired devices

Not all devices use Bluetooth Low Energy (LE)

- Check for other paired device in com.apple.MobileBluetooth.devices.plist or 'bluetoothd' logs

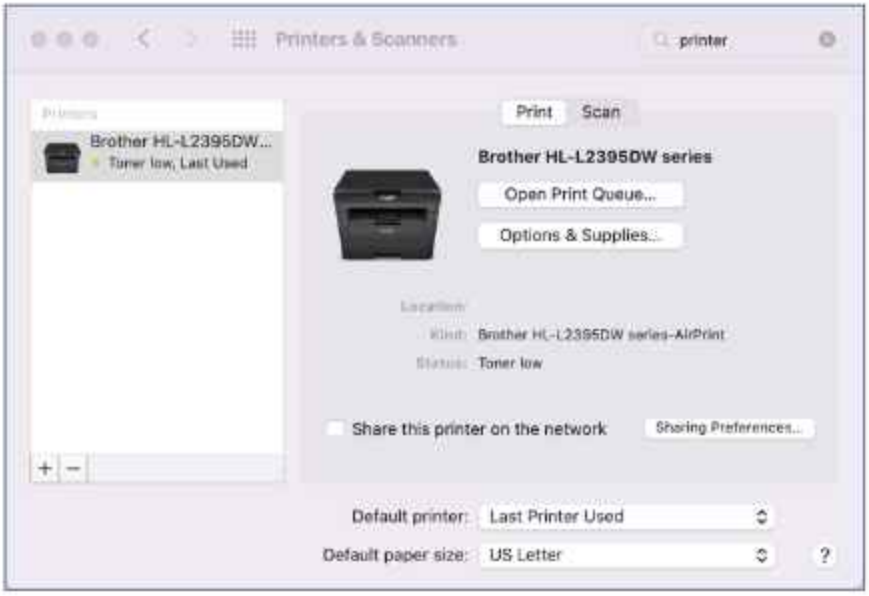
The database, com.apple.MobileBluetooth.ledevices.paired.db contains Bluetooth LE devices that were paired. These MAC addresses do not appear to be randomized.

Not that "Sarah's MBA" appears in both databases ("other" and "paired").



This page intentionally left blank.

Printers & Scanners



The Printers & Scanners Preference Panel can be used to configure local and network printers (and scanners).

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

Root	Array	(1 item)
Item 0	Dictionary	(9 items)
printer-name	String	Brother_HL_L2395DW_series
printer-info	String	Brother HL-L2395DW series
printer-is-accepting-jobs	Boolean	1
printer-location	String	
printer-make-and-model	String	Brother HL-L2395DW series-AirPrint
printer-state	Number	3
printer-state-reasons	Array	(1 item)
Item 0	String	toner-low-warning
printer-type	Number	67,145,812
device-uri	String	dnssd://Brother%20HL-L2395DW%20series._ipp._tcp.local./?uuid=e3248000-80ce-11db-8000-3c2af48cac4f

The printing preference pane contains printer settings for the system. The screenshot shows one printer, a Brother HL-L2395DW, set up on the system. Clues in the `device-uri` key such as "`dnssd`" and "`tcp.local`" can tell us this is a network-based printer (versus local to the system via a cable).

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

Another configuration file, the `printers.conf` file located in `/etc/cups/`, contains specific printer information for each printer installed on the system.

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

macOS Printer Control & Data Files: /private/var/spool/cups [1]

Control Files

- c#######
- Use `strings`

Data Files

- d#######-####
- PDF Files
- Suppose to be removed immediately after successful print

```
root@MBP-M1 cups # pwd
/private/var/spool/cups
root@MBP-M1 cups # ls -l
total 4368
-rw-------   1 root  _lp    6174 Dec 21  2020 c00001
-rw-------   1 root  _lp    6417 Jan 11  2021 c00002
-rw-------   1 root  _lp    6417 Jan 11  2021 c00003
-rw-------   1 root  _lp    5286 Mar 18  2021 c00004
-rw-------   1 root  _lp    6331 Apr  2  2021 c00005
-rw-------   1 root  _lp    5421 Apr  5  2021 c00006
-rw-------   1 root  _lp    5304 Jun  9 08:53 c00007
-rw-------   1 root  _lp    5304 Jun  9 08:55 c00008
-rw-------   1 root  _lp    5304 Jun  9 08:55 c00009
-rw-------   1 root  _lp    5238 Jun  9 08:57 c00010
-rw-------   1 root  _lp    5185 Aug  4 19:53 c00011
-rw-------   1 root  _lp    2494 Aug 11 15:28 c00012
-rw-------   1 root  _lp    5582 Nov 22 12:31 c00013
drwxrwx---  10 root  _lp     320 Nov 22 12:31 cache
-rw-r-----   1 root  _lp  126011 Dec 21  2020 d00001-001
-rw-r-----   1 root  _lp  169790 Jan 11  2021 d00002-001
-rw-r-----   1 root  _lp  169786 Jan 11  2021 d00003-001
-rw-r-----   1 root  _lp  199217 Apr  5  2021 d00006-001
-rw-r-----   1 root  _lp  127359 Jun  9 08:53 d00007-001
-rw-r-----   1 root  _lp  125828 Jun  9 08:54 d00008-001
-rw-r-----   1 root  _lp  127359 Jun  9 08:54 d00009-001
-rw-r-----   1 root  _lp  127359 Jun  9 08:57 d00010-001
-rw-r-----   1 root  _lp  768488 Aug  4 19:53 d00011-001
-rw-r-----   1 root  _lp   31888 Aug 11 15:28 d00012-001
-rw-r-----   1 root  _lp  138878 Nov 22 12:31 d00013-001
drwxrwx--T   5 root  _lp     160 Nov 22 12:31 tmp
```


Each print job has a printer control file located in the `/private/var/spool/cups/` directory. Each file labeled `c#####` correlates with the print job ID. In the example shown above, the system has printed nine documents (`c00001-c00013`). Each control file contains print job metadata.

Each printer control file has a matching printer data file; while the printer control files are persistent, the printer data files are not. These are created at the time of printing and are PDF files. Each is named in line with the printer control jobs (i.e., Printer control file `c00013` would have the data file `d00013-001`).

These files should be removed immediately after a print job has successfully printed, however many times they stick around. If the print job has been canceled or otherwise produced an error, the files may persist for a while longer.

macOS Printer Control & Data Files: /private/var/spool/cups [2]

```
root@MBP-M1 cups # strings c00013
attributes-charset
utf-8H
attributes-natural-language
en-us
printer-uri
2ipp://localhost/printers/Brother_HL_L2395DW_seriesB
job-originating-user-name
oompaB
job-name
'Microsoft is bringing back Clippy - CNNB
```

```
)com.apple.print.JobInfo.PMApplicationName
SafariB
!com.apple.print.JobInfo.PMJobName
'Microsoft is bringing back Clippy - CNNB
"com.apple.print.JobInfo.PMJobOwner
oompaB
+com.apple.print.PageToPaperMappingMediaName
LetterB
```

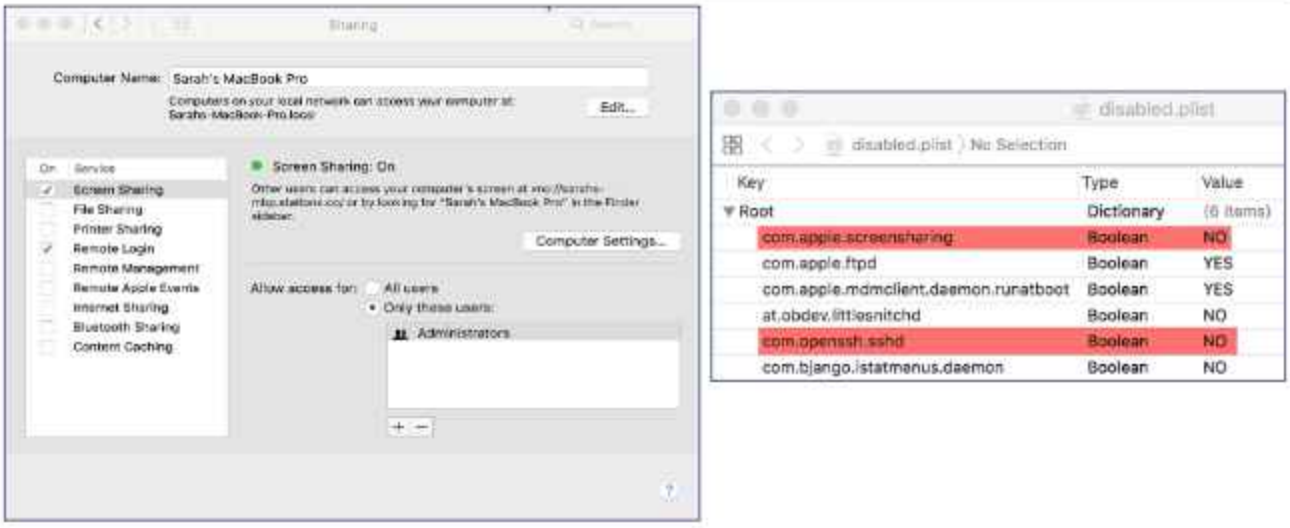
Each printer control file contains metadata about each print job, including which printer was used, originating user account, job name, and which application was used to print from. This information is stored in a proprietary format that is easily viewed using the strings command (although some binary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

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

- The printer is located on the local network "ipp://localhost/printers/Brother_HL_L2395DW_series".
- The originating user account is "oompa".
- The name of the job is "Microsoft is bringing back Clippy - CNNB".
- The application used to print is the Safari web browser.

The data files are PDF files that can be opened using "Preview". The original printer header and footers may also be visible.

macOS Sharing Preferences: /private/var/db/com.apple.xpc.launchd/disabled.plist



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

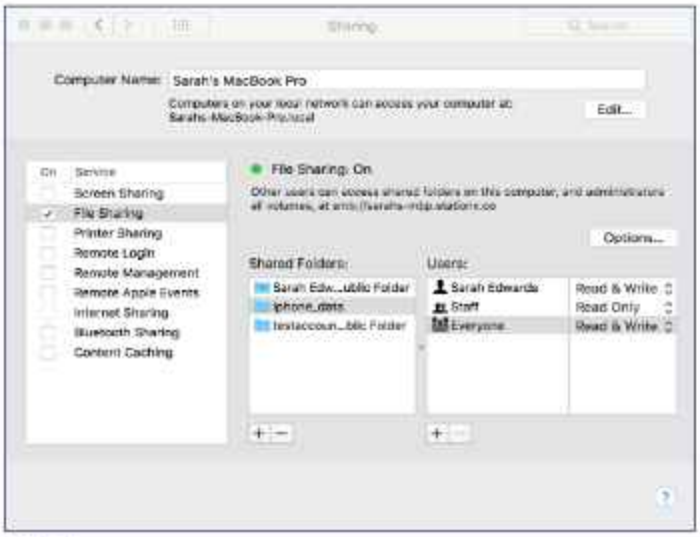
The `overrides.plist/disabled.plist` file contains many configuration keys for various sharing settings. The `com.apple.screensharing` key can contain a Yes (1) or No (0). If Screen Sharing is enabled, the key will show "No" or 0, meaning it is NOT disabled—therefore enabled.

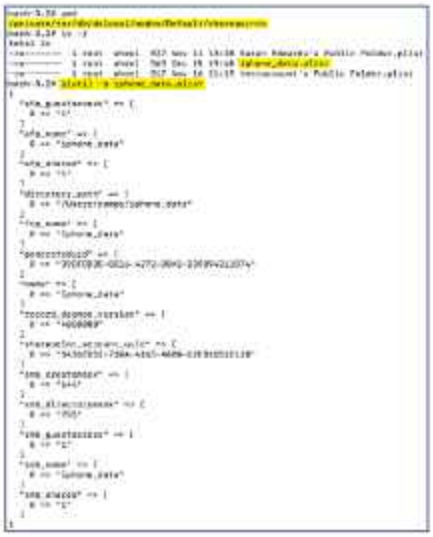
Remote Login allows a user to remotely log in to the system via the SSH (Secure Shell) protocol.

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

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

macOS Sharing Preferences: File Sharing /private/var/db/dslocal/nodes/Default/sharepoints/

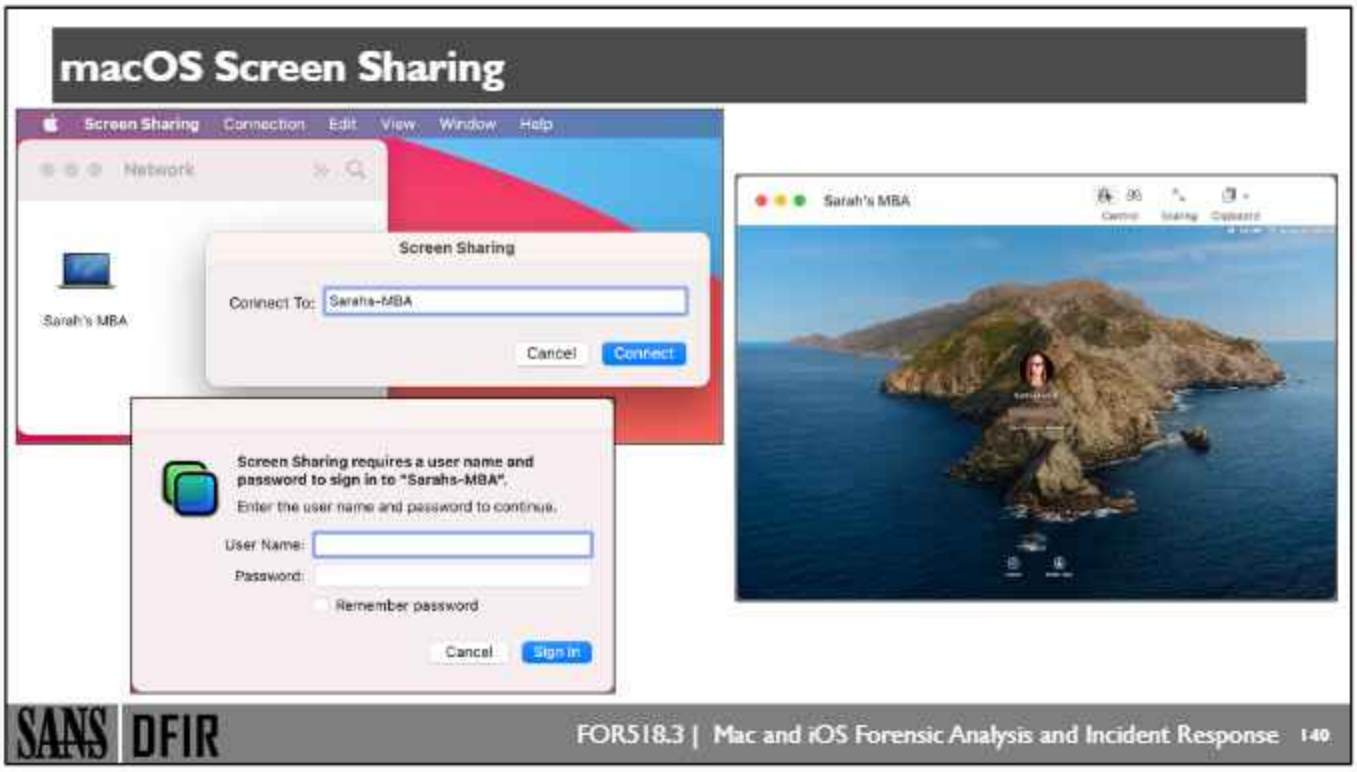




File Sharing allows users to share files on the network or with other users on the system. These shared folders and their metadata can be viewed as plist files in the `/private/var/db/dslocal/nodes/Default/sharepoints/` directory. The example above shows that the folder `iphone_data` was shared along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

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

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



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

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

Users can log into the system using system credentials, by gaining access via another user (i.e., troubleshooting someone else's system), or by turning on a VNC-specific password.

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

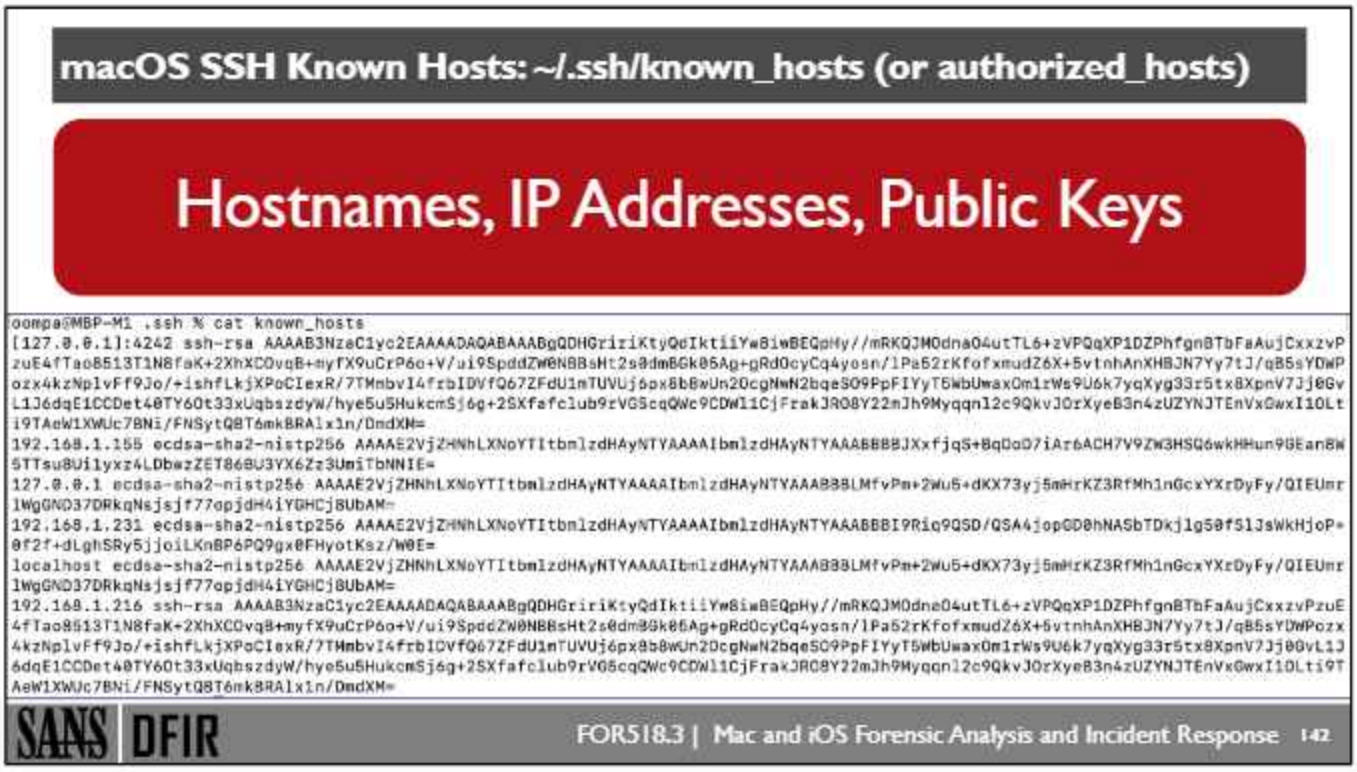

```
nibble:Preferences sledwards$ sudo cat com.apple.VNCSettings.txt
6755221DBA9AF6E2FF1C39567390ADCA
```

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

SANS DFIR FOR518.3 | Mac and iOS Forensic Analysis and Incident Response 141

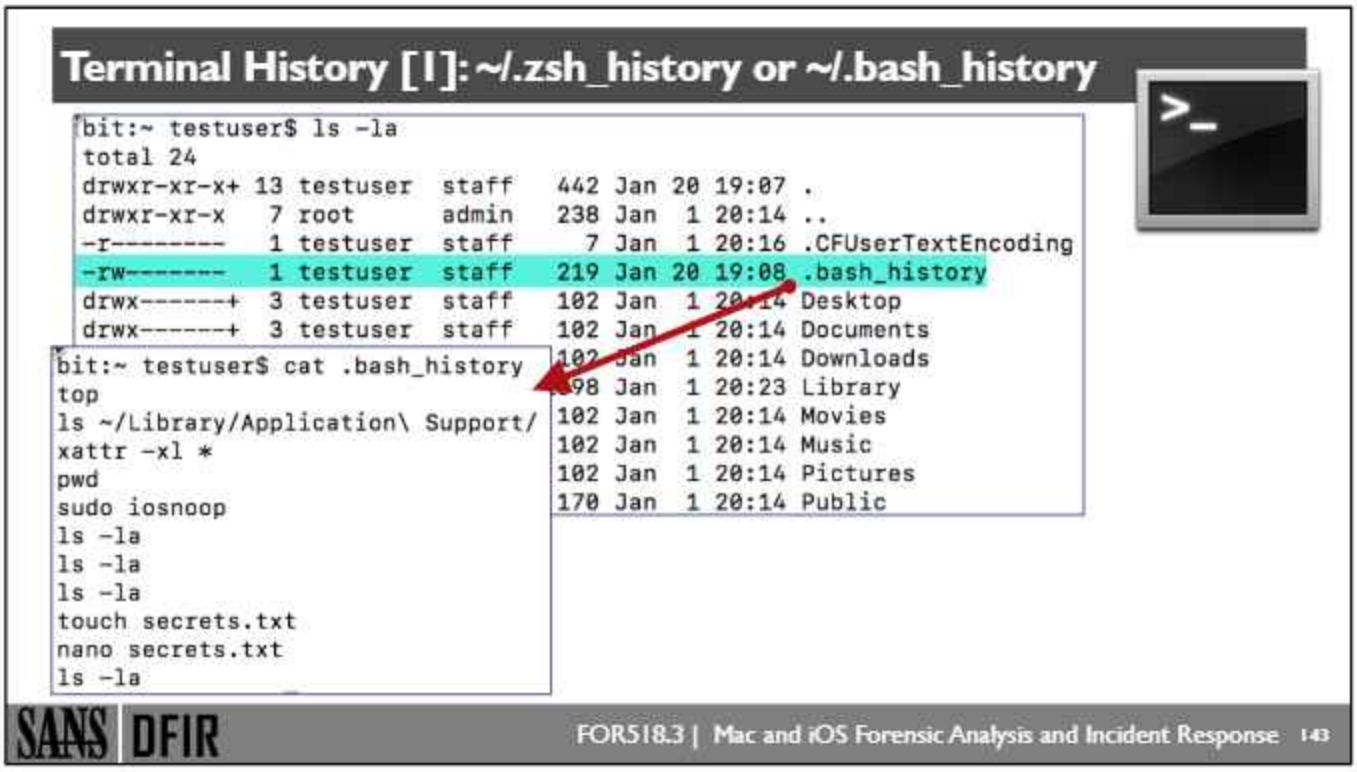
The `com.apple.VNCSettings.txt` text file contains the XOR'ed "encrypted" password to access the system via a VNC viewer client. We can use the Perl script created by Ben Low to "decrypt" the VNC password using the XOR key "`1734516E8BA8C5E2FF1C39567390ADCA`".

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



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

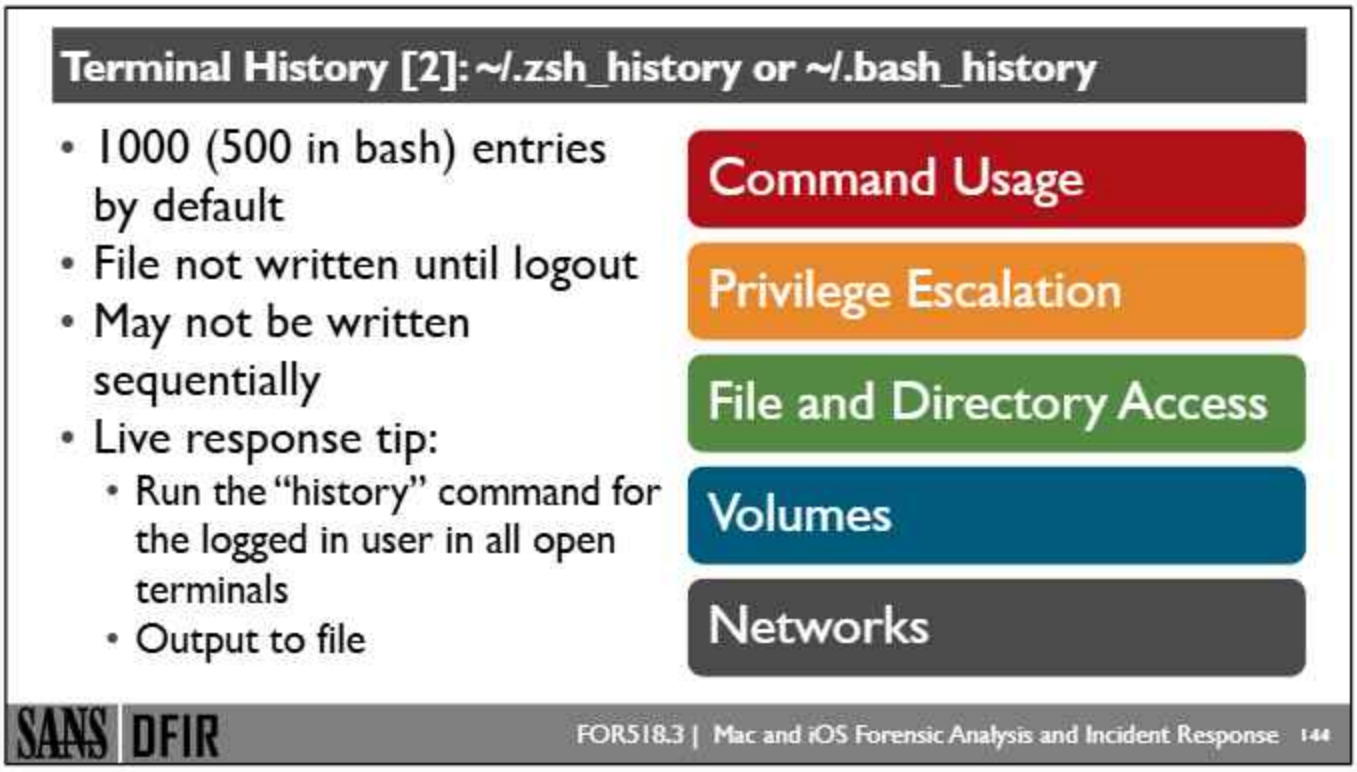
By default, the hostnames and IP address should be human readable. If the user has configured their `/etc/ssh/ssh_config` file to set `HashKnownHosts` to yes, this data will be hashed.



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

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

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



This file contains the command history from the bash/zsh shell. By default, the history file contains a maximum of 1000 (500in bash) commands.

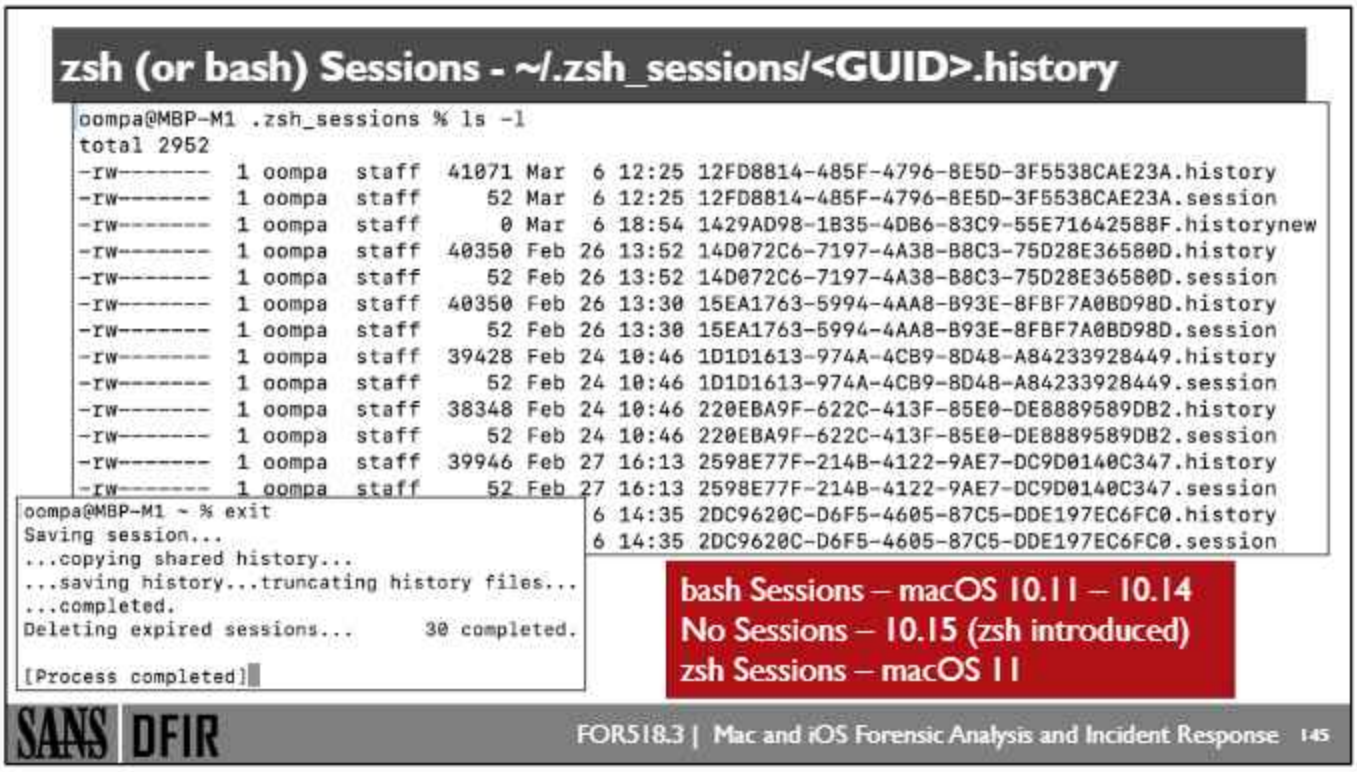
These commands can be useful in showing:

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

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

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

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



Introduced in 10.11 with bash sessions (gone in 10.15 with the introduction of zsh, and back again as zsh sessions in macOS 11). These files have session-based data (which include file system timestamps) for Terminal sessions.

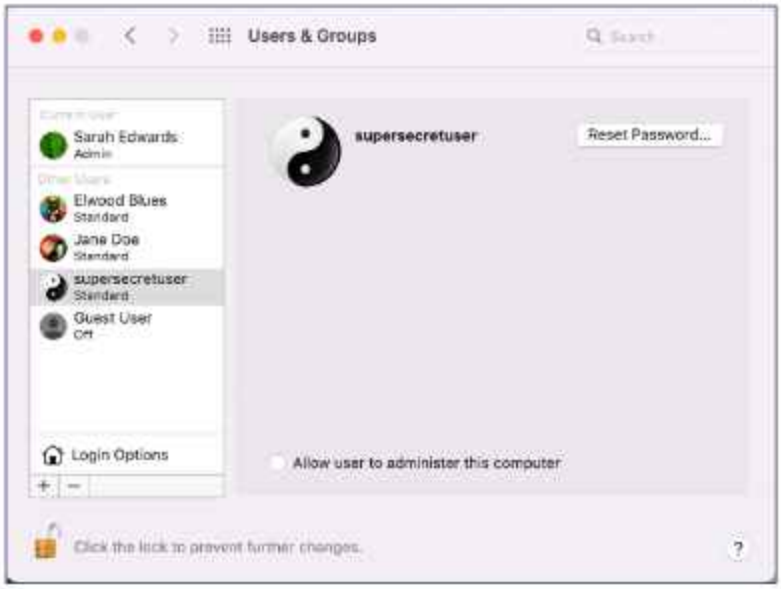
Each `<GUID>.history` file contains the plaintext commands the user performed while in that terminal session. These do not appear to save session history for all time and are at some point removed after a couple weeks.

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

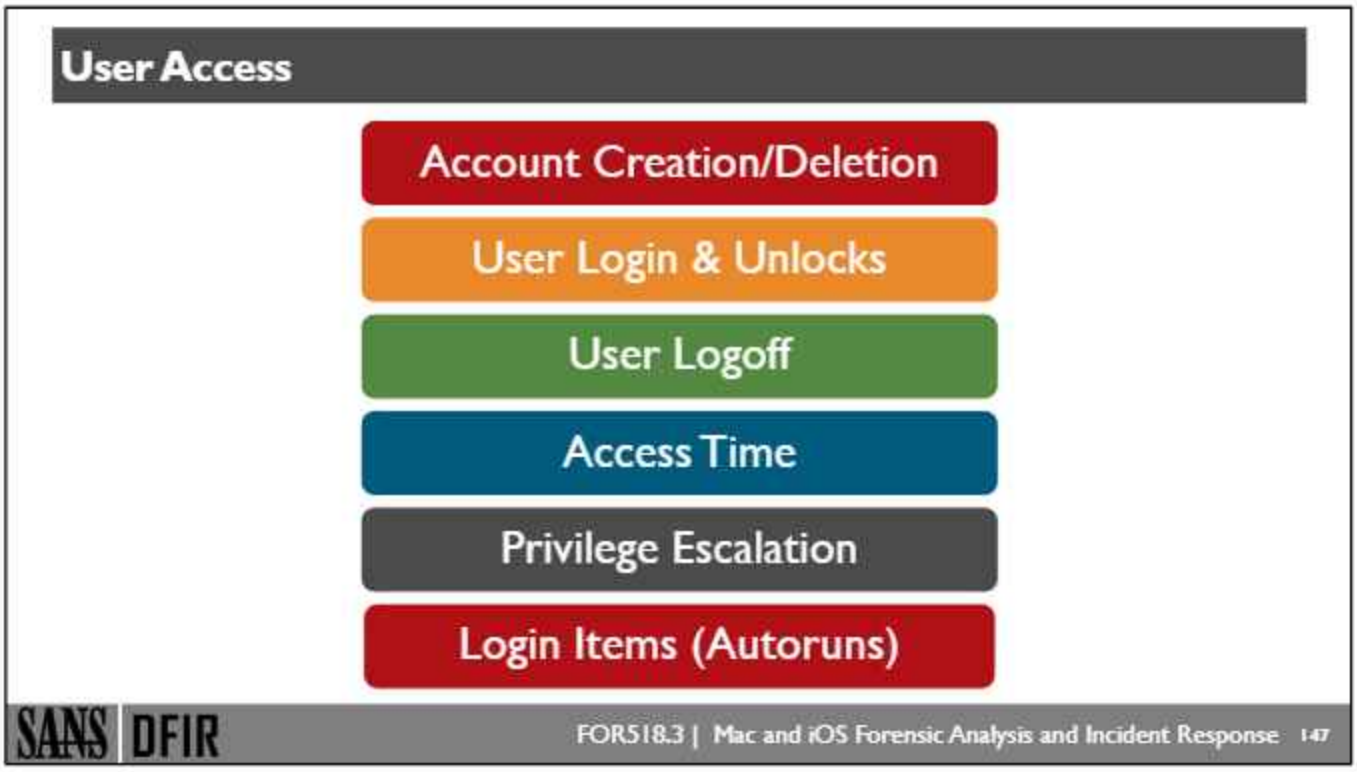
References:
`/etc/bashrc_Apple_Terminal`
`/etc/zshrc_Apple_Terminal`

Users & Groups

- Recall:
  - /private/var/db/dslocal/nodes/Default/users/<user.plist>
  - com.apple.preferences.accounts.plist
  - com.apple.loginwindow.plist



The Users & Groups Preference Panel can be used to configure users of the macOS system.



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

Account Creation (or Deletion) – Audit Files

```
<record version="11" event="SecSrvr AuthEngine" modifier="0" time="Mon Nov 22 18:35:14 2021" msec=" + 463 msec" >
<subject audit-uid="501" uid="501" gid="20" ruid="501" rgid="20" pid="7969" sid="100019" tid="17985 0.0.0.0" />
<text>system.preferences.accounts</text>
<text>client /System/Library/PreferencePanes/Accounts.prefPane/Contents/XPCServices/com.apple.preferences.users.remoteservice.xpc</text>
<text>creator /System/Library/PreferencePanes/Accounts.prefPane/Contents/XPCServices/com.apple.preferences.users.remoteservice.xpc</text>
<return errval="success" retval="0" />
<identity signer-type="1" signing-id="com.apple.authd" signing-id-truncated="no" team-id="" team-id-truncated="no" cdhash="0xfe6b3f256a2c5d
9faf247091d6eb4a6beab77e83" />
</record>
```

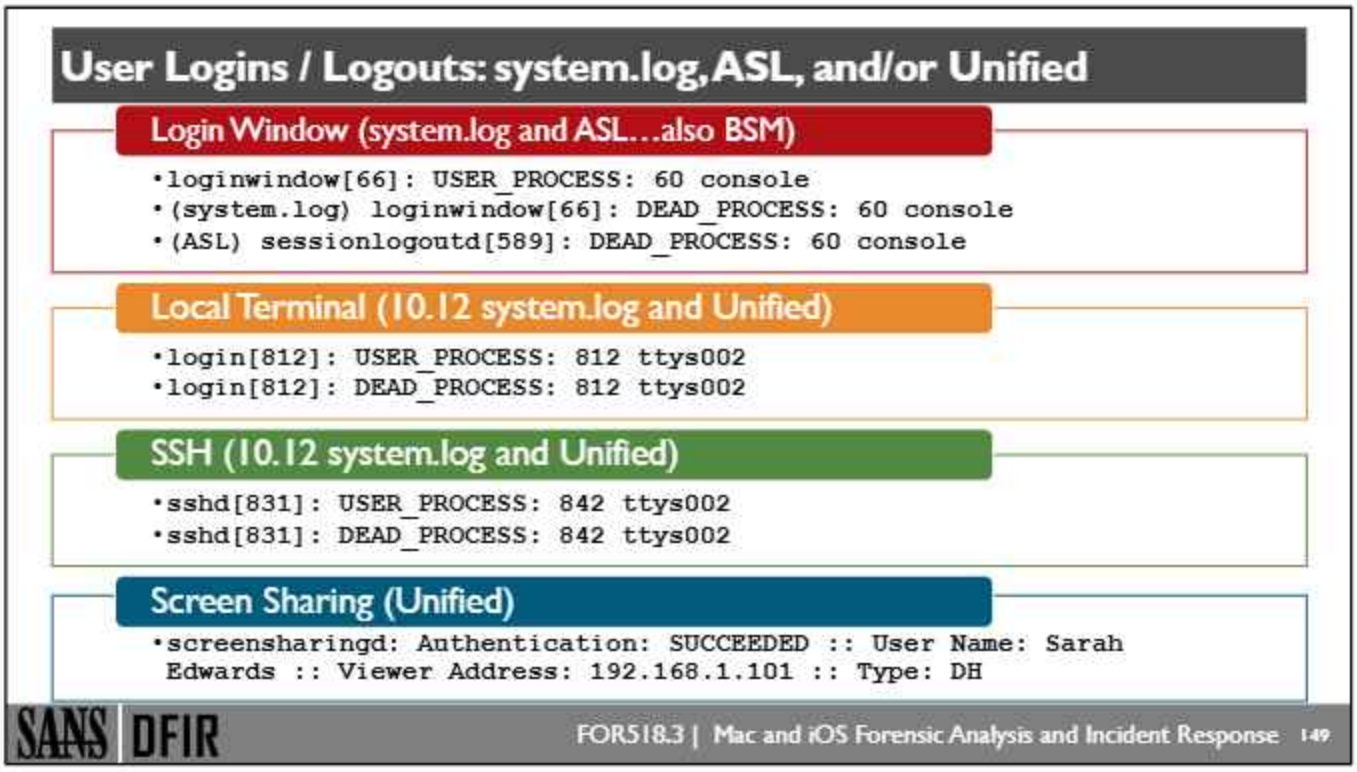
```
<record version="11" event="create user" modifier="0" time="Mon Nov 22 18:36:47 2021" msec=" + 980 msec" >
<subject audit-uid="501" uid="501" gid="20" ruid="501" rgid="20" pid="7969" sid="100019" tid="17985 0.0.0.0" />
<text>Create record type Users &amp;apos;supersecretuser&amp;apos; node &amp;apos;/Local/Default&amp;apos;</text>
<return errval="success" retval="0" />
<identity signer-type="1" signing-id="com.apple.opendirectoryd" signing-id-truncated="no" team-id="" team-id-truncate
d="no" cdhash="0x0c7b8da500d433862cf599c97182fc4c71f33d7e" />
</record>
```

User account creation can also be of immense interest to an investigator. Audit records for user creation are very verbose. Lots of related entries are created.

One event is for the authentication to the Users and Groups Preference Panel.

Another event, is for the "`create user`". This entry includes the following data:

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



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

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

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

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

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

The Login Window example shows a login (PID 74) and logoff (PID 60) for two different sessions, labeled as "`console`". Note this type uses the "`loginwindow`" process.

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

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

Screen Sharing logins show an entry like the one shown in the example above. It records the username (may be Full Name or username) and the IP address where the connection is coming from.

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

References:
https://www.mac4n6.com/blog/2020/4/26/analysis-of-apple-unified-logs-quarantine-edition-entry-4-its-login-week
https://www.mac4n6.com/blog/2020/4/30/analysis-of-apple-unified-logs-quarantine-edition-entry-6-working-from-home-remote-logins

macOS Screen Unlocks and User Login Types

- `log show --predicate 'eventMessage contains "com.apple.sessionagent.screenIs"'`

- `log show --predicate 'eventMessage contains "LWScreenLockAuthentication" and (eventMessage contains "| Verifying" or eventMessage contains "| Using")'`

1. Regular Password
2. Touch ID
3. Apple Watch

SANS DFIR | FOR518.3 | Mac and iOS Forensic Analysis and Incident Response | 151

Looking for messages that contain '`com.apple.sessionagent.screenIs`' string. This is going to show if the system is locked or unlocked, and which user is currently logged in with their user ID (UID). These are not technically logins since the user is already logged in but are useful for telling if the screen is locked or not.

- `com.apple.sessionagent.screenIsLocked` = Screen is Locked
- `com.apple.sessionagent.screenIsUnlocked` = Screen is Unlocked

While knowing if the screen is locked or unlocked is good, sometimes you may want to know how a macOS system was unlocked. We can do this by looking for '`LWSCreenLockAuthentication`' entries that contain '`| Verifying`' or '`| Using`'. Using the text as hints, we can determine if it was via password, TouchID, or by their Apple Watch.

Regular Password:

- "Verifying using PAM configuration screensaver"

TouchID:

- "Using localAuthentication hints"
- "Using hint-provided username oompa"
- "Verifying using PAM configuration screensaver_la"

Auto Unlock with Apple Watch:

- "Using continuity hints"
- "Using hint-provided username oompa"
- "Verifying using PAM configuration screensaver_aks"

Reference:
https://www.mac4n6.com/blog/2020/4/26/analysis-of-apple-unified-logs-quarantine-edition-entry-4-its-login-week

```
xxxxxMacxxx-MBA ~ % log show --last 4h --predicate 'eventMessage contains "com.apple.sessionagent.screenIs"'
log: warning: ./system_logs.logarchive present but reading from system log store.
Filtering the log data using "composedMessage CONTAINS "com.apple.sessionagent.screenIs""
Skipping info and debug messages, pass --info and/or --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2020-04-25 16:05:14.497971-0400 0xc9d6     Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[SessionAgentNotificationCenter sntNotifyShxredSpace:key:toValue:] | setNatifyShxredSpace: com.apple.sessionagent.screenIsLocked, to value:1 using token:82
2020-04-25 16:05:16.669893-0400 0xc9d6     Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[SessionAgentNotificationCenter sendSSDNotification:forUserID:] | sendSSDNotification: com.apple.sessionagent.screenIsLocked, with userID:585
2020-04-25 16:23:58.657833-0400 0x3d2      Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[SessionAgentNotificationCenter sntNotifyShxredSpace:key:toValue:] | setNatifyShxredSpace: com.apple.sessionagent.screenIsLocked, to value:0 using token:82
2020-04-25 16:23:58.657129-0400 0x342      Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[SessionAgentNotificationCenter sendSSDNotification:forUserID:] | sendSSDNotification: com.apple.sessionagent.screenIsUnlocked, with userID:585
2020-04-25 16:23:58.8901E5-0400 0x1a73c    Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[SessionAgentNotificationCenter sntNotifyShxredSpace:key:toValue:] | setNatifyShxredSpace: com.apple.sessionagent.screenIsLocked, to value:1 using token:82
2020-04-25 16:23:59.8901C9-0400 0x1a73c    Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[SessionAgentNotificationCenter sendSSDNotification:forUserID:] | sendSSDNotification: com.apple.sessionagent.screenIsLocked, with userID:585
2020-04-25 16:55:56.8909Y1-0400 0x342      Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[SessionAgentNotificationCenter sntNotifyShxredSpace:key:toValue:] | setNatifyShxredSpace: com.apple.sessionagent.screenIsLocked, to value:0 using token:82
2020-04-25 16:55:56.8910B1-0400 0x342      Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[SessionAgentNotificationCenter sendSSDNotification:forUserID:] | sendSSDNotification: com.apple.sessionagent.screenIsUnlocked, with userID:585
-------------------------------------------------------------------------------------------------------------------------------------------
Log      - Default:          0, Info:            0, Debug:          0, Error:          0, Fault:         0
Activity - Create:           0, Transition:      0, Actions:        0
```

```
xxxxxMacxxx-MBA ~ % log show --info --debug  --last 4m --predicate 'eventMessage contains "LWScreenLockAuthentication" and (eventMessage contains "| Verifying" or eventMessage contains "| Using")'
log: warning: ./system_logs.logarchive present but reading from system log store.
Filtering the log data using "composedMessage CONTAINS "LWScreenLockAuthentication" AND (composedMessage CONTAINS "| Verifying" OR composedMessage CONTAINS "| Using")"
Timestamp                       Thread     Type        Activity             PID    TTL
2020-04-25 20:30:57.795168-0400 0x257e7    Default     0x2f5ab              174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[LWScreenLockAuthentication _authWithUsername:password:userContinuityMint:localAuthenticationMints:] | Verifying using PAM configuration screensaver
2020-04-25 20:30:58.337485-0400 0x255e2    Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[LWScreenLockAuthentication _authWithUsername:password:userContinuityMint:localAuthenticationMints:] | Using localAuthentication Mints
2020-04-25 20:30:58.337418-0400 0x255e2    Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[LWScreenLockAuthentication _authWithUsername:password:userContinuityMint:localAuthenticationMints:] | Using hint-provided username scope
2020-04-25 20:30:58.723374-0400 0x255e2    Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[LWScreenLockAuthentication _authWithUsername:password:userContinuityMint:localAuthenticationMints:] | Verifying using PAM configuration screensaver_ia
2020-04-25 20:31:12.854543-0400 0x25ae3    Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[LWScreenLockAuthentication _authWithUsername:password:userContinuityMint:localAuthenticationMints:] | Using continuity Mints
2020-04-25 20:31:12.854484-0400 0x25ae3    Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[LWScreenLockAuthentication _authWithUsername:password:userContinuityMint:localAuthenticationMints:] | Using hint-provided username scope
2020-04-25 20:31:12.954143-0400 0x25ae3    Default     0x0                  174    5    loginwindow [com.apple.loginwindow.logging(Standard)] -[LWScreenLockAuthentication _authWithUsername:password:userContinuityMint:localAuthenticationMints:] | Verifying using PAM configuration screensaver_aut
-------------------------------------------------------------------------------------------------------------------------------------------
Log      - Default:          7, Info:            0, Debug:          4, Error:          0, Fault:         0
Activity - Create:           0, Transition:      0, Actions:        0
```

Privilege Escalation – sudo and su

- `log show --predicate 'process == "sudo" and eventMessage contains "TTY="'`

```
compa@Sarahs-MBA ~ % log show --last 1d --predicate 'process == "sudo" and eventMessage contains "TTY="'
log: warning: ./system_logs.logarchive present but reading from system log store.
Filtering the log data using "process == "sudo" AND composedMessage CONTAINS "TTY="
Skipping info and debug messages, pass --info and/or --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2020-04-21 17:35:15.481168-0400 0x43c897   Default     0x0                  15763  0    sudo: compa : TTY=ttys000 ; PWD=/Users/compa ; USER=root ; COMMAND=/usr/bin/iosreco
2020-04-21 17:35:48.017291-0400 0x43c9aa   Default     0x0                  15883  0    sudo: compa : TTY=ttys000 ; PWD=/Users/compa ; USER=root ; COMMAND=/usr/bin/fs_usage
2020-04-21 17:41:13.214252-0400 0x43d469   Default     0x0                  16149  0    sudo: compa : TTY=ttys000 ; PWD=/Users/compa ; USER=root ; COMMAND=/bin/mkdir /Volumes/boring_mount_point
2020-04-21 17:42:04.495254-0400 0x43d6a4   Default     0x0                  16220  0    sudo: compa : TTY=ttys000 ; PWD=/Users/compa ; USER=root ; COMMAND=/usr/bin/fs_usage

Log      - Default:     4, Info:          0, Debug:         0, Error:         0, Fault:         0
Activity - Create:      0, Transition:    0, Actions:       0
```

- `log show --predicate '(process == "su" or process == "sudo") and eventMessage contains "tty"'`

```
compa@Sarahs-MBA ~ % log show --last 20m --info --debug --predicate '(process == "su" or process == "sudo") and eventMessage contains "tty"'
log: warning: ./system_logs.logarchive present but reading from system log store.
Filtering the log data using "(process == "su" OR process == "sudo") AND composedMessage CONTAINS "tty""
Timestamp                       Thread     Type        Activity             PID    TTL
2020-04-22 17:22:02.778148-0400 0x4a211a   Default     0x0                  66728  0    sudo:    compa : 2 incorrect password attempts ; TTY=ttys001 ; PWD=/Users/compa ; USER=root ; COMMAND=/bin/zsh
2020-04-22 17:24:04.417471-0400 0x4a2471   Default     0x0                  66824  0    sudo:    compa : TTY=ttys001 ; PWD=/Users/compa ; USER=root ; COMMAND=/bin/zsh
2020-04-22 17:28:26.429451-0400 0x4a2fa9   Default     0x0                  67102  0    su: BAD SU compa to janedoe on /dev/ttys002
2020-04-22 17:28:41.836300-0400 0x4a3068   Default     0x0                  67128  0    su: BAD SU compa to janedoe on /dev/ttys003
2020-04-22 17:28:47.788114-0400 0x4a30c8   Default     0x0                  67144  0    su: BAD SU compa to janedoe on /dev/ttys003
2020-04-22 17:28:55.133251-0400 0x4a3155   Default     0x0                  67160  0    su: compa to janedoe on /dev/ttys003

Log      - Default:     4, Info:          0, Debug:         0, Error:         0, Fault:         0
Activity - Create:      0, Transition:    0, Actions:       0
```

153

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

The `sudo` example on the top shows successful commands using `sudo`. These entries include:

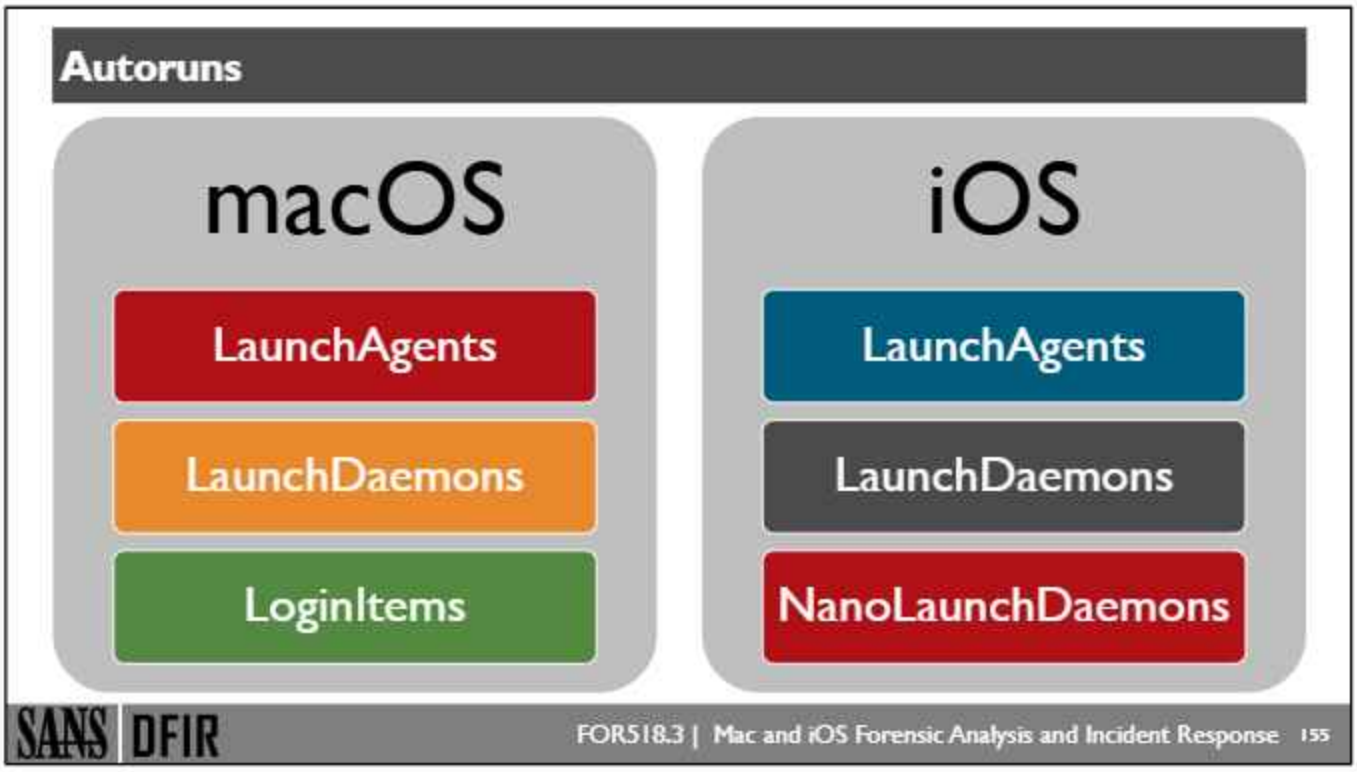
- Terminal window
- Current working directory
- Escalated user account
- Command

On the bottom, the user attempted to use `sudo` (`sudo -s`) twice and had the incorrect password before finally providing the correct password. Following this, the user attempted to `su` to janedoe incorrectly three times before again providing the correct password.

Reference:
https://www.mac4n6.com/blog/2020/4/22/analysis-of-apple-unified-logs-quarantine-edition-entry-2-sudo-make-me-a-sandwich

```
oompa@Sarahs-MBA ~ %  log show --last 1d --predicate 'process == "sudo" and eventMessage contains "TTY="'
log: warning: ./system_logs.logarchive present but reading from system log store.
Filtering the log data using "process == "sudo" AND composedMessage CONTAINS "TTY=""
Skipping info and debug messages, pass --info and/or --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2020-04-21 17:35:15.481168-0400 0x43c897   Default     0x0                  25763  0    sudo:    oompa : TTY=ttys000 ; PWD=/Users/oompa ; USER=root ; COMMAND=/usr/bin/iosnoop
2020-04-21 17:35:40.017291-0400 0x43c9aa   Default     0x0                  25803  0    sudo:    oompa : TTY=ttys000 ; PWD=/Users/oompa ; USER=root ; COMMAND=/usr/bin/fs_usage
2020-04-21 17:41:13.214252-0400 0x43d460   Default     0x0                  26149  0    sudo:    oompa : TTY=ttys000 ; PWD=/Users/oompa ; USER=root ; COMMAND=/bin/mkdir /Volumes/boring_mount_point
2020-04-21 17:42:04.495254-0400 0x43d6a4   Default     0x0                  26220  0    sudo:    oompa : TTY=ttys000 ; PWD=/Users/oompa ; USER=root ; COMMAND=/usr/bin/fs_usage

Log      - Default:          4, Info:               0, Debug:            0, Error:          0, Fault:          0
Activity - Create:           0, Transition:         0, Actions:          0
```

```
oompa@Sarahs-MBA ~ % log show --last 20m --info --debug --predicate '(process == "su" or process == "sudo") and eventMessage contains "tty"'
log: warning: ./system_logs.logarchive present but reading from system log store.
Filtering the log data using "(process == "su" OR process == "sudo") AND composedMessage CONTAINS "tty""
Timestamp                       Thread     Type        Activity             PID    TTL
2020-04-22 17:22:52.778148-0400 0x4a2116   Default     0x0                  66720  0    sudo:    oompa : 2 incorrect password attempts ; TTY=ttys001 ; PWD=/Users/oompa ; USER=root ; COMMAND=/bin/zsh
2020-04-22 17:24:04.417471-0400 0x4a2471   Default     0x0                  66824  0    sudo:    oompa : TTY=ttys001 ; PWD=/Users/oompa ; USER=root ; COMMAND=/bin/zsh
2020-04-22 17:28:25.429451-0400 0x4a2fa9   Default     0x0                  67182  0    su: BAD SU oompa to janedoe on /dev/ttys002
2020-04-22 17:28:41.335398-0400 0x4a3068   Default     0x0                  67128  0    su: BAD SU oompa to janedoe on /dev/ttys003
2020-04-22 17:28:47.780114-0400 0x4a30c8   Default     0x0                  67144  0    su: BAD SU oompa to janedoe on /dev/ttys003
2020-04-22 17:28:55.133251-0400 0x4a3156   Default     0x0                  67160  0    su: oompa to janedoe on /dev/ttys003

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



macOS and iOS systems have a variety of locations that applications can auto start from.

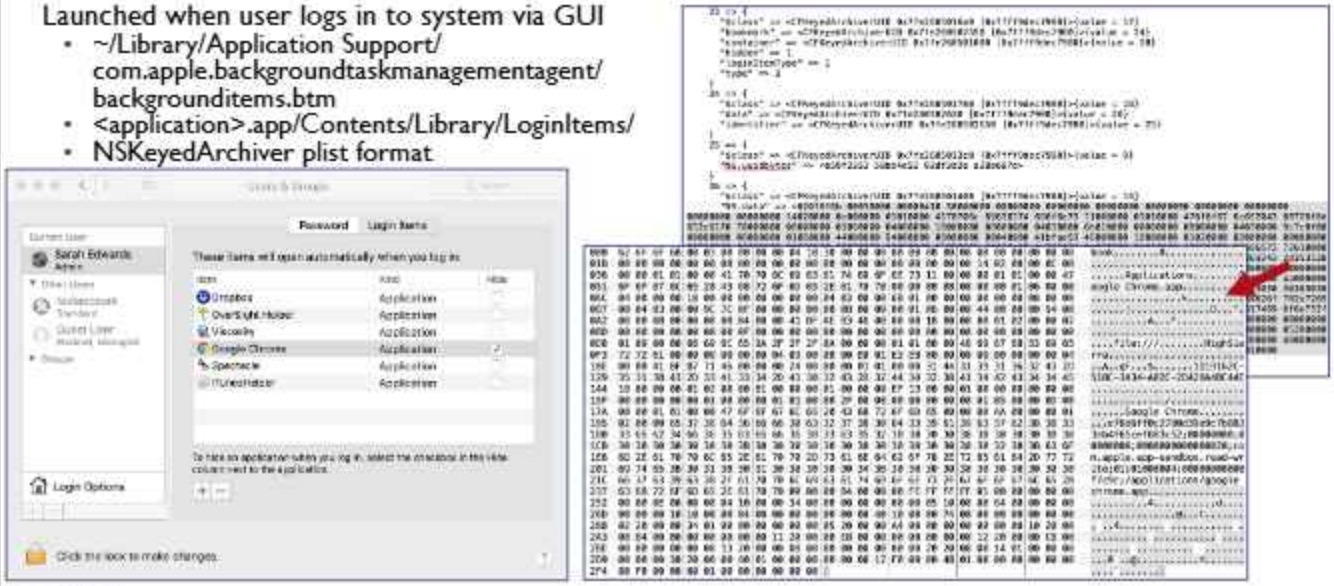
While these locations can be used legitimately, malware authors also tend to use them—especially on macOS.

A tool you might want to look at to find autoruns and auditing information is Knockknock from Objective-See: https://objective-see.com/products/knockknock.html.

Autoruns: macOS Login Items – 10.13+

Launched when user logs in to system via GUI

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



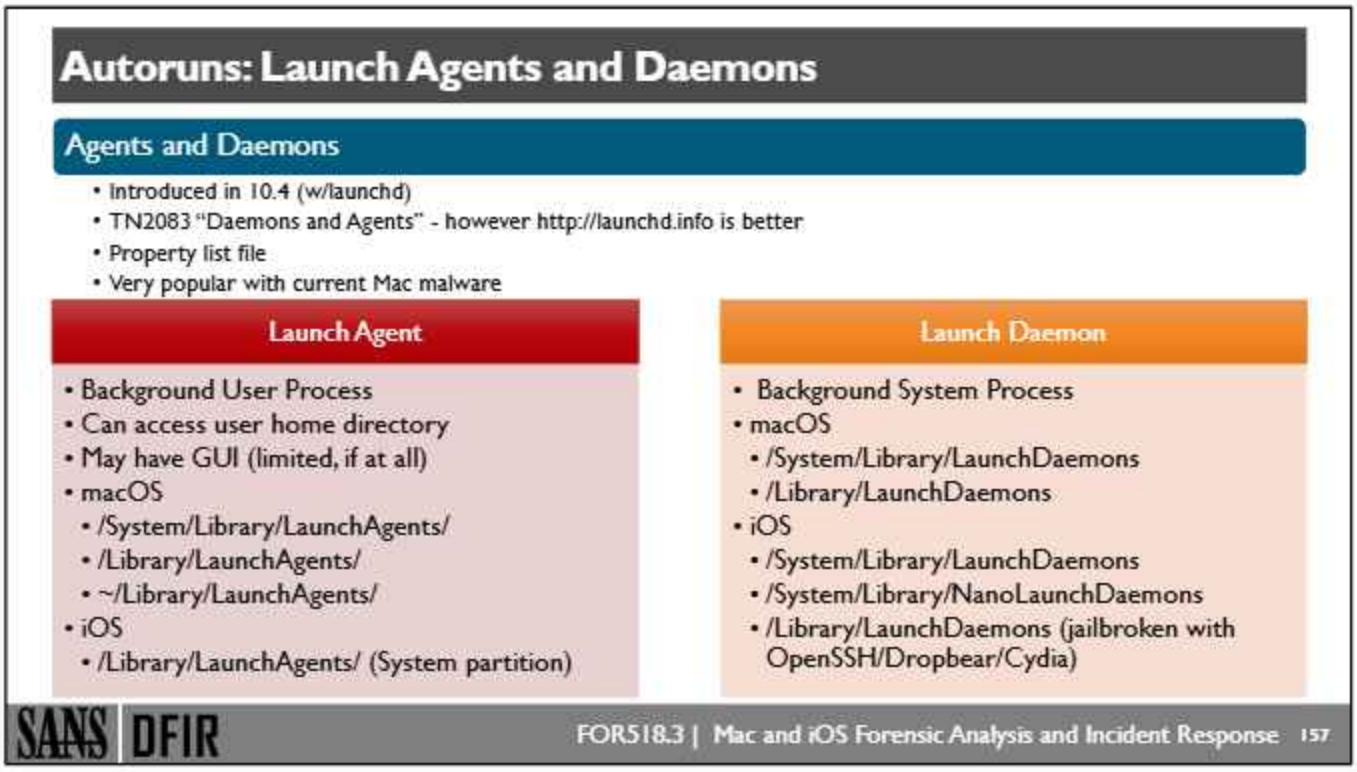
Each user may have a set of autoruns that get launched when a user logs in via the system GUI login. Each login item can be "hidden" on launch, which means the login item is hidden from view when it is starting up. `Login items`, while usually legitimate, may be used as a malware persistence mechanism.

This is an NSKeyedArchiver plist file. Each user will have their own login items.

Locations:

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

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



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

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

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

Reference:

Daemons and Agents, Tech Note 2083 (TN2083)

https://developer.apple.com/library/archive/technotes/tn2083/_index.html

Autoruns: Launch Agent Examples [1]

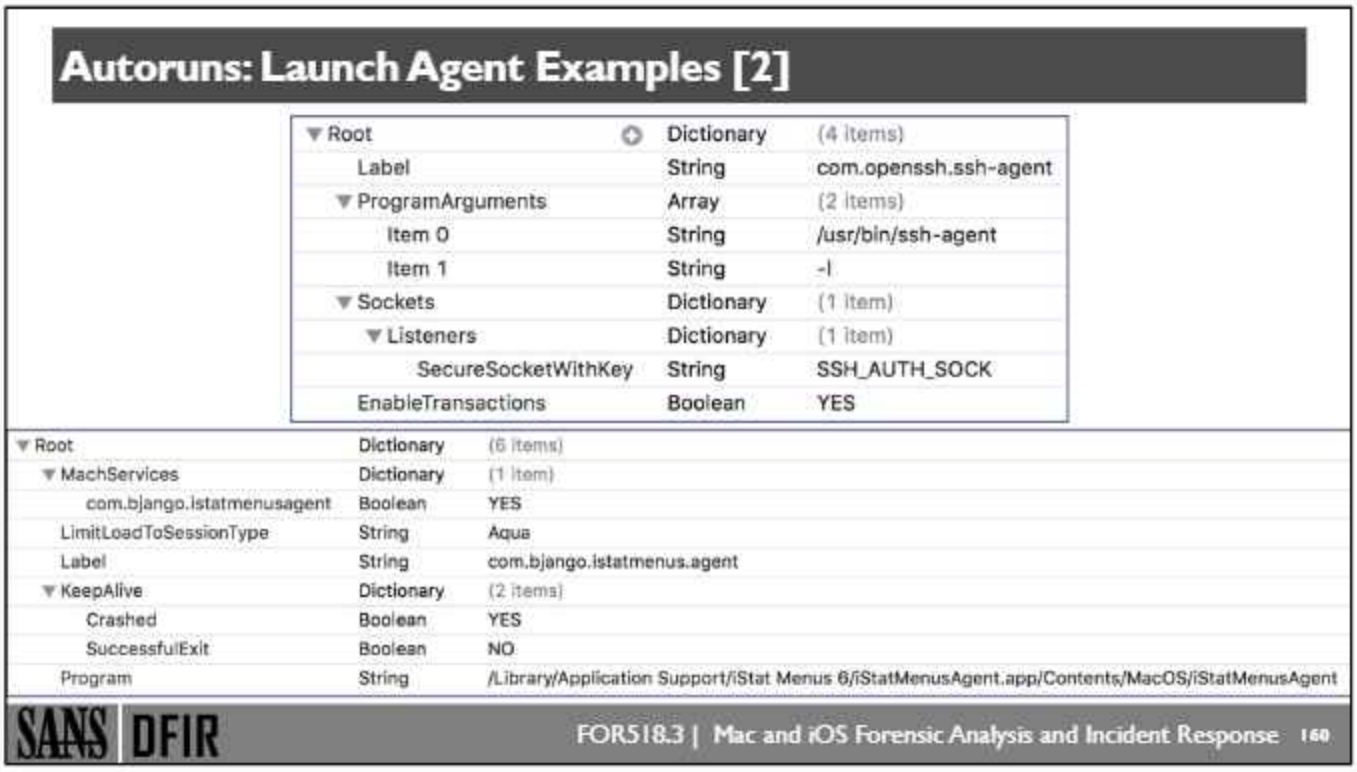
```
oompa@MBP-M1 /etc % ls -l /Library/LaunchAgents
total 40
-rw-r--r--  1 root  wheel  463 Nov 24 10:59 et.obdev.littlesnitch.agent.plist
-rw-r--r--  1 root  wheel  674 Dec 18 09:09 com.bjango.istatmenus.agent.plist
-rw-r--r--  1 root  wheel  682 Dec 18 09:09 com.bjango.istatmenus.status.plist
-rw-r--r--  1 root  wheel  523 Nov 23 09:04 com.microsoft.OneDriveStandaloneUpdater.plist
-rw-r--r--  1 root  wheel  377 Feb 19 15:32 com.microsoft.update.agent.plist
oompa@MBP-M1 /etc % ls -l ~/Library/LaunchAgents
total 40
-rw-r--r--  1 oompa  staff  685 Nov 23 10:05 com.dropbox.DropboxMacUpdate.agent.plist
-rw-r--r--@ 1 oompa  staff  889 Mar  4 19:59 com.google.keystone.agent.plist
-rw-r--r--@ 1 oompa  staff  915 Mar  4 19:59 com.google.keystone.xpcservice.plist
-rw-r--r--  1 oompa  staff  455 Feb  1 06:02 com.logmein.GoToMeeting.G2MAIRUploader.plist
-rw-r--r--  1 oompa  staff  450 Feb  1 06:02 com.logmein.GoToMeeting.G2MUpdate.plist
oompa@MBP-M1 /etc % ls -l /System/Library/LaunchAgents/

total 2568
-rw-r--r--  1 root  wheel   656 Jan  1  2020 com.apple.AMPArtworkAgent.plist
-rw-r--r--  1 root  wheel   764 Jan  1  2020 com.apple.AMPDeviceDiscoveryAgent.plist
-rw-r--r--  1 root  wheel   716 Jan  1  2020 com.apple.AMPDevicesAgent.plist
-rw-r--r--  1 root  wheel   763 Jan  1  2020 com.apple.AMPLibraryAgent.plist
-rw-r--r--  1 root  wheel   737 Jan  1  2020 com.apple.AMPSystemPlayerAgent.plist
-rw-r--r--  1 root  wheel  1062 Jan  1  2020 com.apple.AOSHeartbeat.plist
-rw-r--r--  1 root  wheel   839 Jan  1  2020 com.apple.AOSPushRelay.plist
-rw-r--r--  1 root  wheel   660 Jan  1  2020 com.apple.AccessibilityVisualsAgent.plist
-rw-r--r--  1 root  wheel   613 Jan  1  2020 com.apple.AddressBook.AssistantService.plist
-rw-r--r--  1 root  wheel   744 Jan  1  2020 com.apple.AddressBook.ContactsAccountsService.plist
-rw-r--r--  1 root  wheel  1063 Jan  1  2020 com.apple.AddressBook.SourceSync.plist
-rw-r--r--  1 root  wheel   677 Jan  1  2020 com.apple.AddressBook.abd.plist
```


The screenshot above shows a typical example of the various `LaunchAgents` directories.

Each launch agent property list is named in the reverse DNS format. Generally, if you are looking for a malicious Launch Agent you may choose to look at the filename for odd sounding items or looking at the filesystem timestamps for the one that fits your malicious timeline.

If all else fails, usually a good internet search for the filename may provide information whether it is legitimate or not.

```
oompa@MBP-M1 /etc % ls -l /Library/LaunchAgents
total 40
-rw-r--r--  1 root  wheel  463 Nov 24 10:59 at.obdev.littlesnitch.agent.plist
-rw-r--r--  1 root  wheel  674 Dec 18 09:09 com.bjango.istatmenus.agent.plist
-rw-r--r--  1 root  wheel  682 Dec 18 09:09 com.bjango.istatmenus.status.plist
-rw-r--r--  1 root  wheel  523 Nov 23 09:04 com.microsoft.OneDriveStandaloneUpdater.plist
-rw-r--r--  1 root  wheel  377 Feb 19 15:32 com.microsoft.update.agent.plist
oompa@MBP-M1 /etc % ls -l ~/Library/LaunchAgents
total 40
-rw-r--r--  1 oompa  staff  685 Nov 23 10:05 com.dropbox.DropboxMacUpdate.agent.plist
-rw-r--r--@ 1 oompa  staff  809 Mar  4 19:59 com.google.keystone.agent.plist
-rw-r--r--@ 1 oompa  staff  915 Mar  4 19:59 com.google.keystone.xpcservice.plist
-rw-r--r--  1 oompa  staff  455 Feb  1 06:02 com.logmein.GoToMeeting.G2MAIRUploader.plist
-rw-r--r--  1 oompa  staff  450 Feb  1 06:02 com.logmein.GoToMeeting.G2MUpdate.plist
oompa@MBP-M1 /etc % ls -l /System/Library/LaunchAgents/

total 2568
-rw-r--r--  1 root  wheel   656 Jan  1  2020 com.apple.AMPArtworkAgent.plist
-rw-r--r--  1 root  wheel   754 Jan  1  2020 com.apple.AMPDeviceDiscoveryAgent.plist
-rw-r--r--  1 root  wheel   716 Jan  1  2020 com.apple.AMPDevicesAgent.plist
-rw-r--r--  1 root  wheel   763 Jan  1  2020 com.apple.AMPLibraryAgent.plist
-rw-r--r--  1 root  wheel   737 Jan  1  2020 com.apple.AMPSystemPlayerAgent.plist
-rw-r--r--  1 root  wheel  1082 Jan  1  2020 com.apple.AOSHeartbeat.plist
-rw-r--r--  1 root  wheel   839 Jan  1  2020 com.apple.AOSPushRelay.plist
-rw-r--r--  1 root  wheel   680 Jan  1  2020 com.apple.AccessibilityVisualsAgent.plist
-rw-r--r--  1 root  wheel   613 Jan  1  2020 com.apple.AddressBook.AssistantService.plist
-rw-r--r--  1 root  wheel   744 Jan  1  2020 com.apple.AddressBook.ContactsAccountsService.plist
-rw-r--r--  1 root  wheel  1063 Jan  1  2020 com.apple.AddressBook.SourceSync.plist
-rw-r--r--  1 root  wheel   677 Jan  1  2020 com.apple.AddressBook.abd.plist
```



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

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

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

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

Reference:
`launchd.plist` man Page

Autoruns: Launch Daemon Example

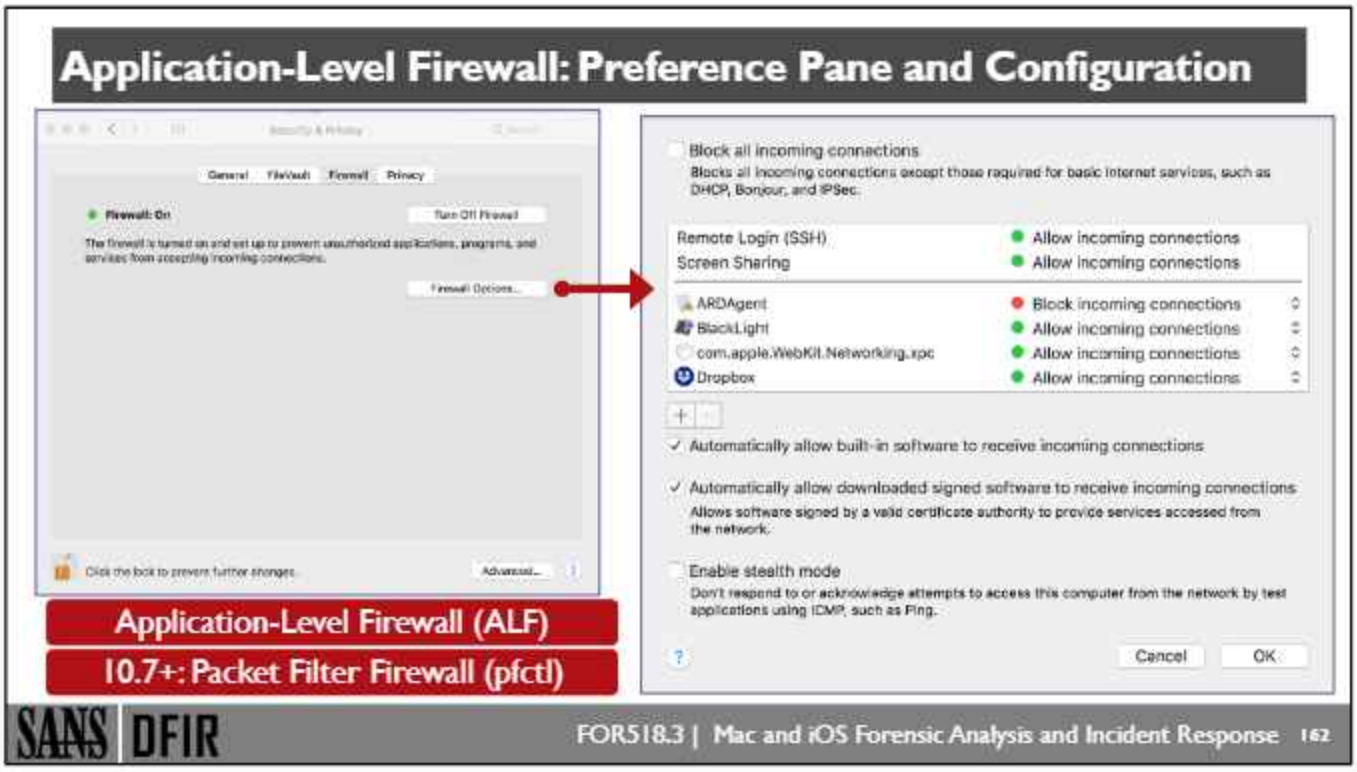
```
Sat Dec  3 09:46:21 EST 2017

Removing old temporary files:

Clearing out old system announcements:

Removing stale files from /var/rwho:

Disk status:
Filesystem    Size   Used  Avail Capacity  iused               ifree %iused  Mounted on
/dev/disk1s1  932Gi  96Gi  269Gi    27% 2178221 9223372036849913650    0%   /
/dev/disk1s4  932Gi  3.0Gi 269Gi     1%       5 9223372036854775802    0%   /private/var/vm
/dev/disk2s3  95Mi   82Mi   48Mi    64%     922          4291466317    0%   /Volumes/iExplorer

Network interface status:
Name   Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16384 <Link#1>                         53436     0    53436     0     0
lo0   16384 127           localhost          53436     -    53436     -     -
lo0   16384 localhost   ::1                  53436     -    53436     -     -
lo0   16384 fe80::1%lo0 fe80::1              53436     -    53436     -     -
gif0* 1280  <Link#2>                             0     0        0     0     0
stf0* 1280  <Link#3>                             0     0        0     0     0
XHC20 0     <Link#4>                             0     0        0     0     0
XHC21 0     <Link#5>                             0     0        0     0     0
XHC1* 0     <Link#6>                             0     0        0     0     0
en0   1500  <Link#7>    f4:0f:24:3b:a2:3b 19149842     0 49451953     0     0
en0   1500  fe80::1cde: fe80:8::1cde:81a6 19149842     - 49451953     -     -
en0   1500  10.11.12/24   sarahs-mbp.stat 19149842     - 49451953     -     -
p2p0  2304  <Link#9>    0a:0f:24:3b:a2:3b        0     0        0     0     0
awdl0 1484  <Link#10>   9a:7c:72:84:53:7d       91     0       93     0     0
awdl0 1484  fe80::9c7b: fe80:a::9c7b:727f       91     -       93     -     -
en1   1500  <Link#11>   ac:87:a3:0a:55:1a        0     0        0     0     0
en2   1500  <Link#12>   ac:87:a3:0a:55:1b        0     0        0     0     0
en3   1500  <Link#13>   ac:87:a3:0a:55:1c        0     0        0     0     0
en4   1500  <Link#14>   ac:87:a3:0a:55:1d        0     0        0     0     0
bridg 1500  <Link#15>   ae:87:a3:0a:55:1a        0     0        0     0     0
utun0 2000  <Link#16>                           0     0        2     0     0
utun1 1380  fe80::2818: fe80:10::2818:e1b        0     -        2     -     -
utun2 1380  <Link#17>                          74     0       50     0     0
utun3 1380  fe80::e8a8: fe80:11::e8a8:55b       74     -       50     -     -
utun4 1380  <Link#18>                         182     0       40     0     0
utun5 1380  fe80::c023: fe80:12::c023:ce5     182     -       40     -     -
en0   1500  <Link#7>    ac:de:48:00:11:22     1534     0     1615     0     0
en0   1500  fe80::aede: fe80:7::aede:48ff     1534     -     1615     -     -

Local system status:
 9:46  up 3 days, 19:38, 1 user, load averages: 89.78 28.48 11.78

-- End of daily output --
```

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

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

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

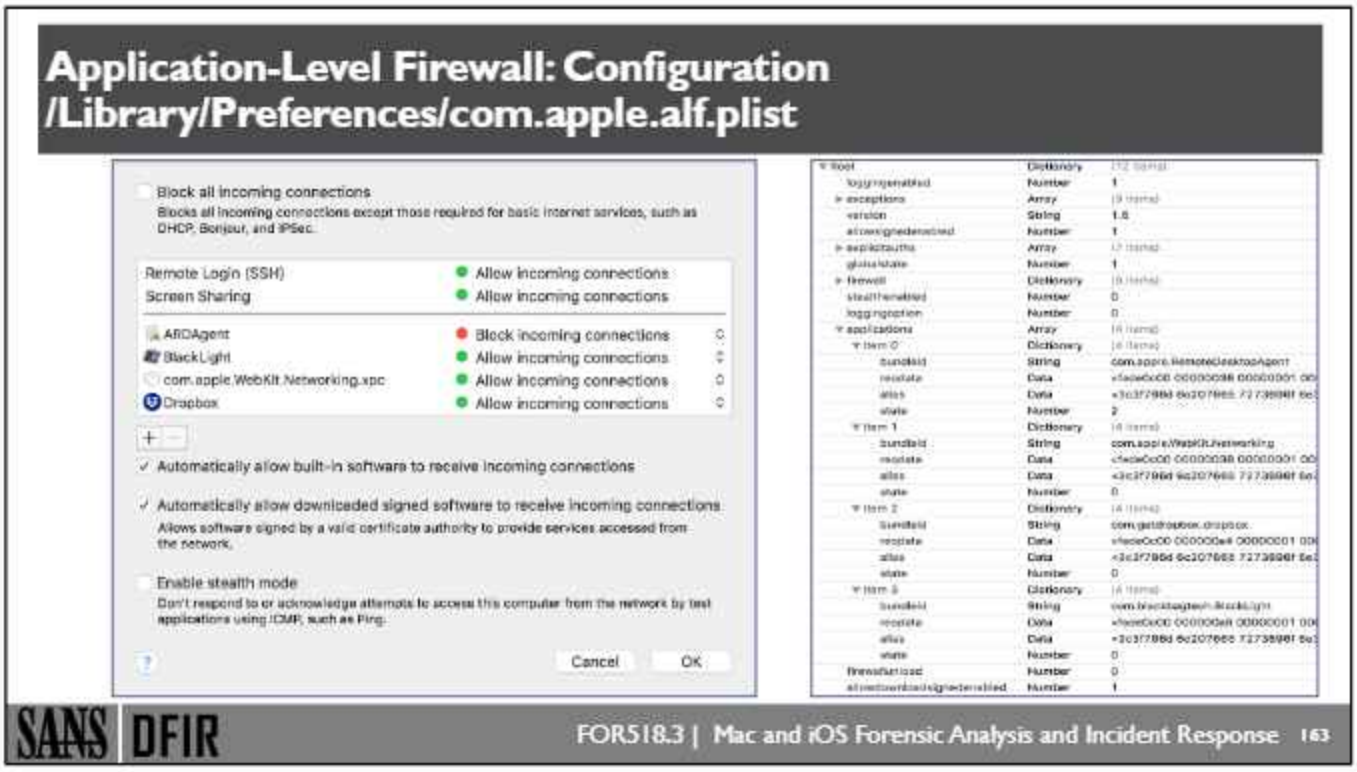
The property list also contains the key `LaunchEvents` keys. This shows how often to run the process; in this case, every day (86,400 seconds), with a grace period of four hours (14,400 seconds).



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

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

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



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

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

- 1 = Firewall Enabled
- 0 = Firewall Disabled

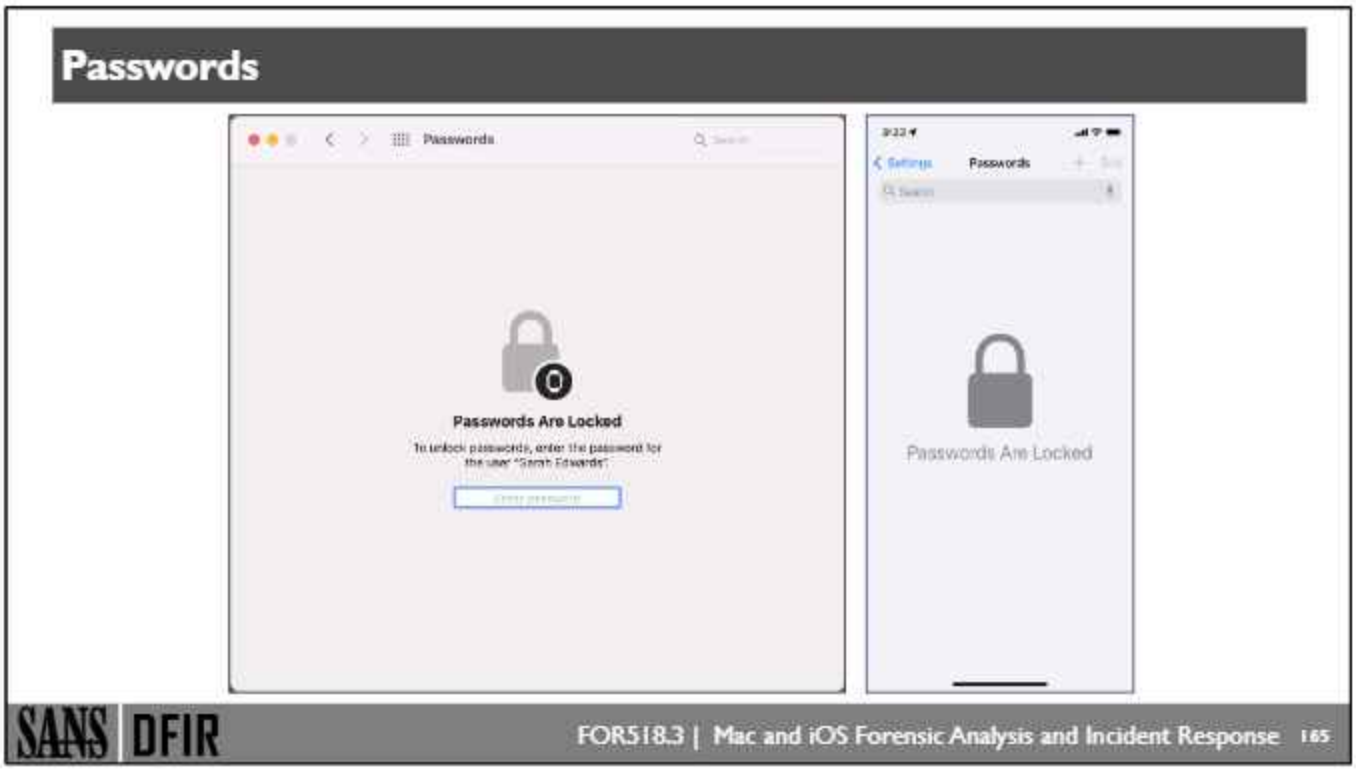
In the screenshot of the application firewall, there are two checkboxes the users can configure. The following keys hold this configuration:

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

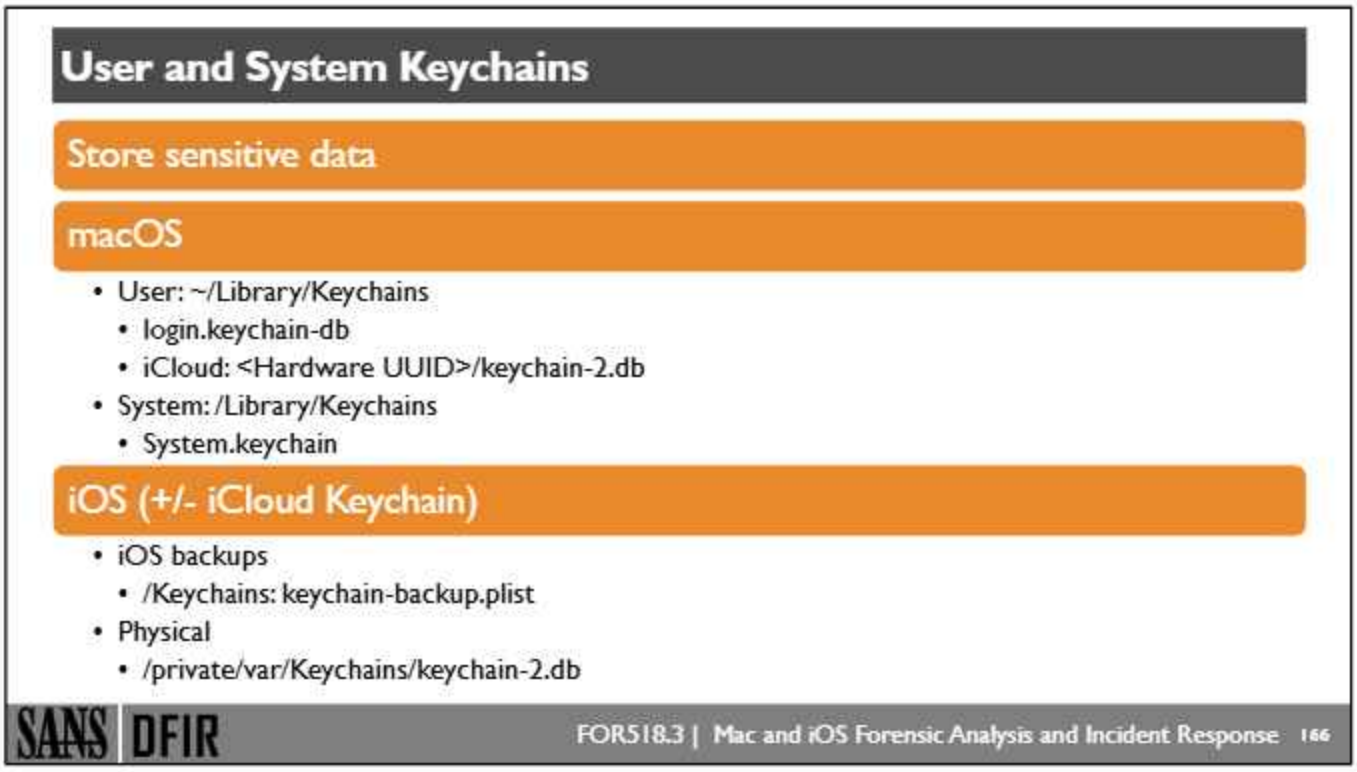
The `applications` key lists the applications configured in the firewall, as shown in the screenshot on the previous slide. Each application item contains the applications bundle ID, alias data, and the state. A state of "0" allows incoming connections, while a state of "2" blocks incoming connections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

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

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



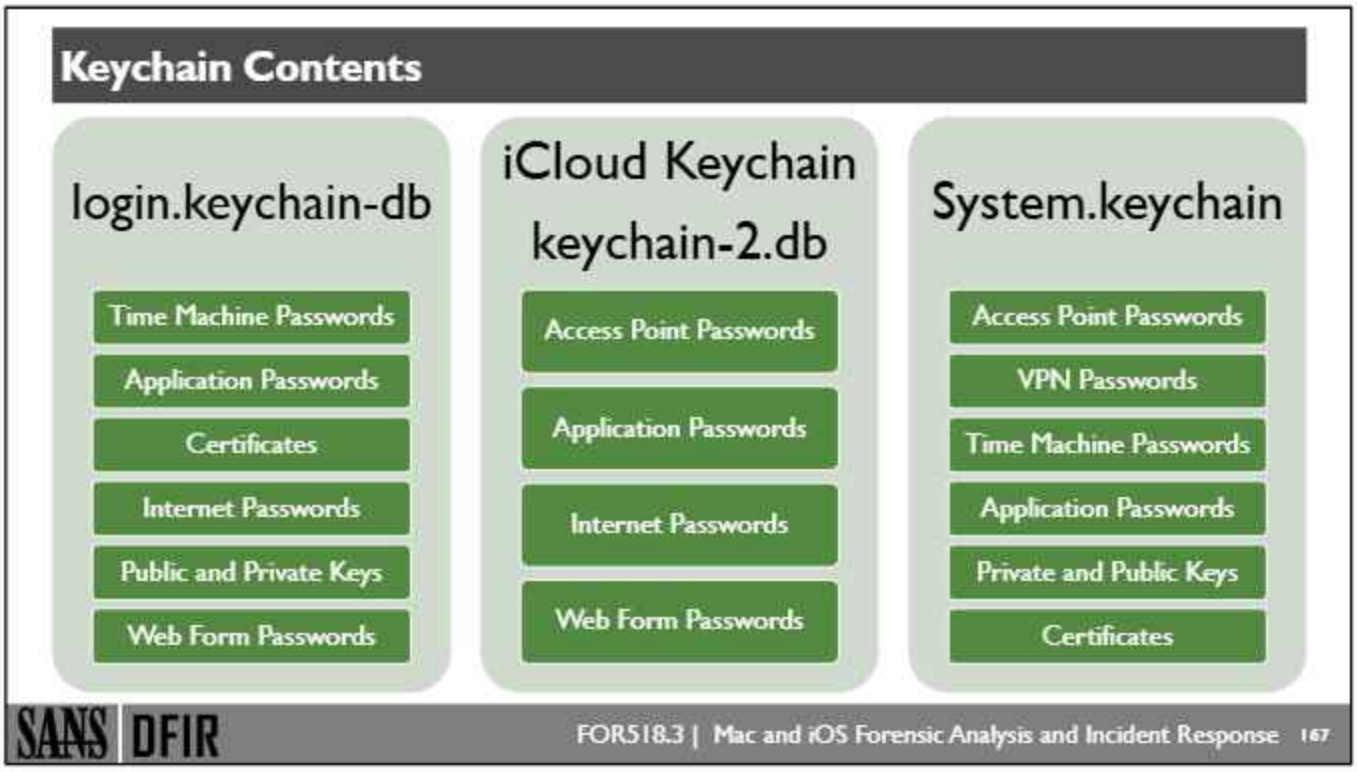
The Passwords Preference Panel can be used to view stored secrets that are stored in the Keychains.



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

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

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



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

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

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

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

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

Keychain Disk Layout: macOS

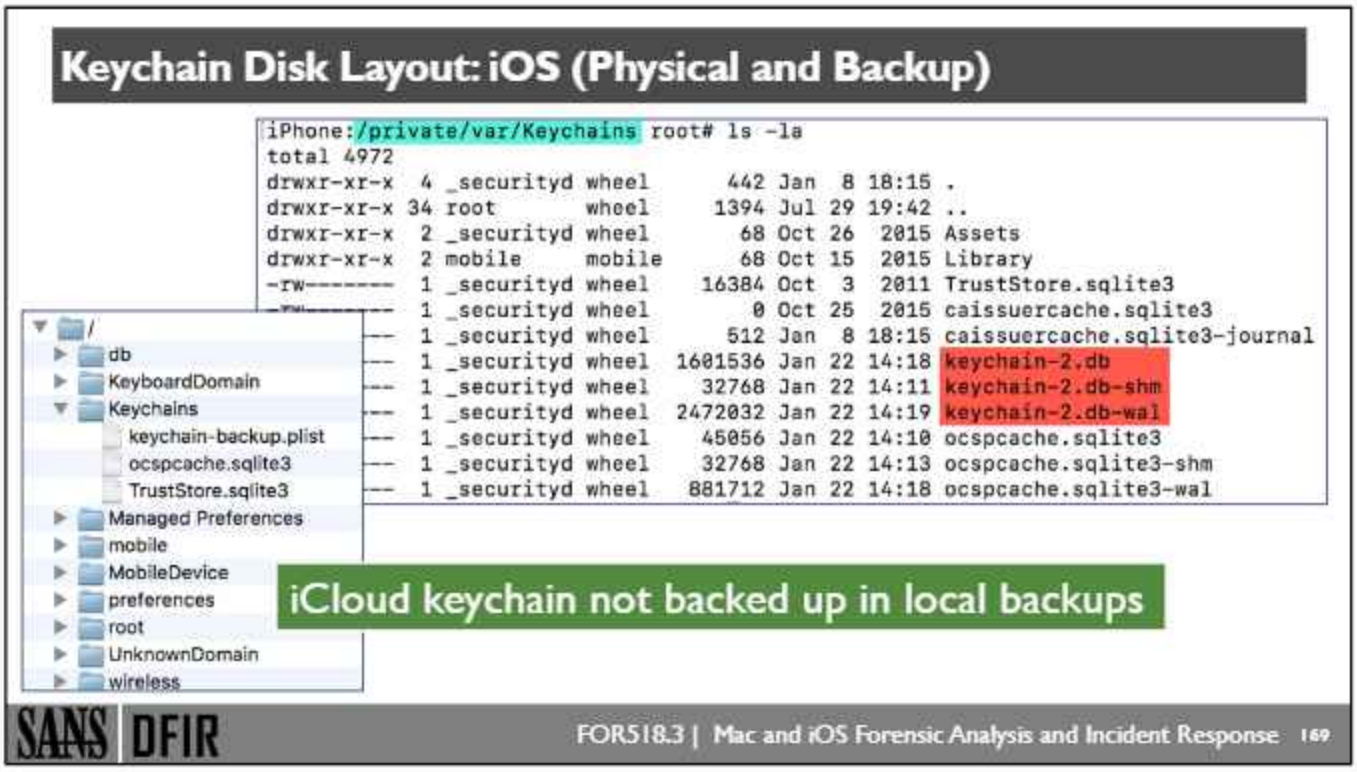
```
bit:Keychains oompa$ pwd
/Users/oompa/Library/Keychains
bit:Keychains oompa$ ls -lR
total 1328
drwx------  11 oompa  staff     374 Dec 30 10:01 502A32AD-3CAF-5585-BC09-053E285901C8
-rw-r--r--@  1 oompa  staff  650440 Jan 22 14:14 login.keychain-db
-rw-------   1 oompa  staff   27176 Dec 26 11:16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

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


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

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

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

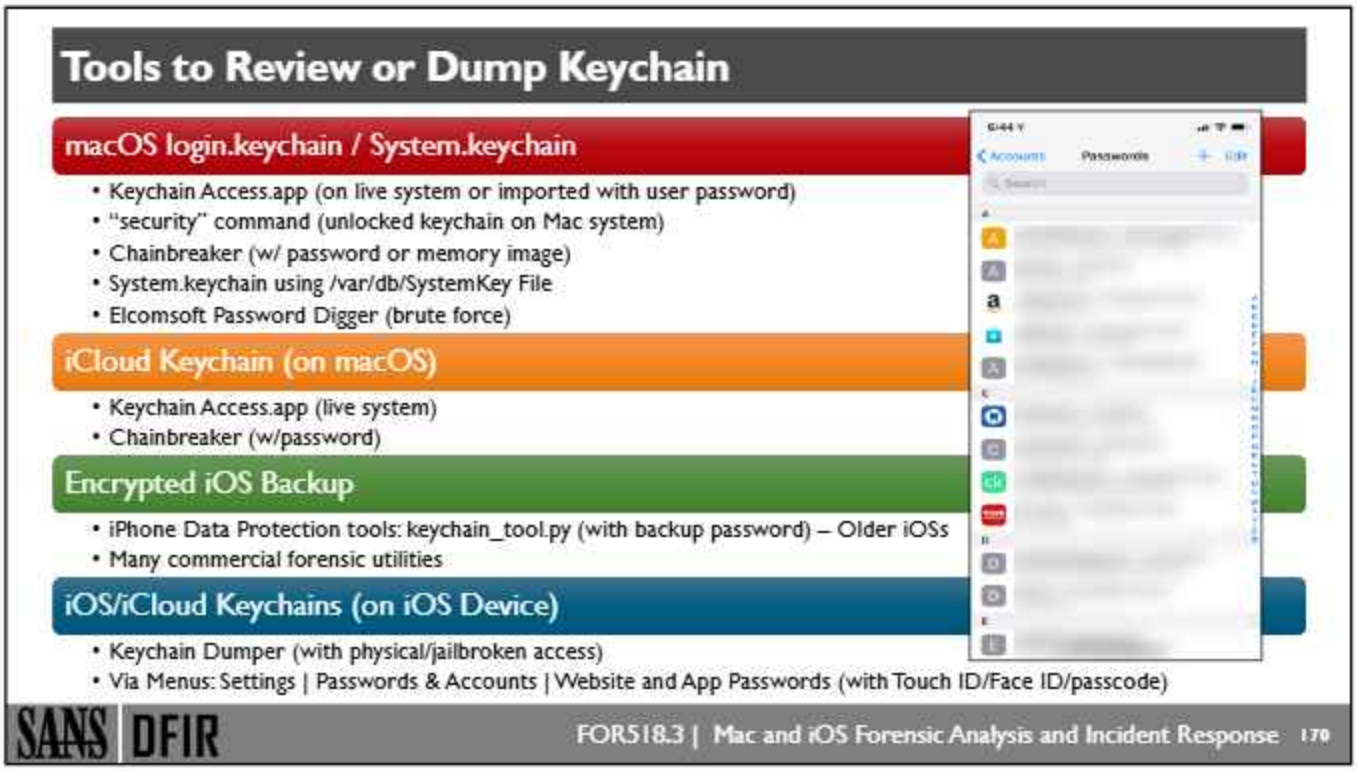
The `System.keychain` file is stored in the System Library (`/Library/Keychains/` directory).



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

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

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



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

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

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

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

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

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

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

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

References:
System.keychain via StackExchange: https://apple.stackexchange.com/questions/307189/how-to-decrypt-the-system-keychain-from-another-mac-at-the-command-line
iPhone Data Protection Tools: https://github.com/dinosec/iphone-dataprotection
Chainbreaker: https://github.com/n0fate/chainbreaker
Elcomsoft Password Digger: https://www.elcomsoft.com/epd.html
Keychain Dumper: https://github.com/ptoomey3/Keychain-Dumper

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

Note: The default password for a user's `login.keychain` is their login password; however, this can be changed if the user has the option to change it.

The analyst can select the "`Show password`" checkbox and input the user's keychain password (by default, the login password).

Other Keychain Access

"strings *" keychain

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

"security" command

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

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

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

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

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

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

macOS: `/var/folders/<darwin_user_dir>/0/com.apple.ScreenTimeAgent/`
iOS: `/private/var/mobile/Library/Application Support/com.apple.remotemanagementd/`

Screen Time is an application that keeps track of time spent in various apps, application notifications, and 'pickups'. (A pickup may be when a user picks up their device after receiving an app notification.).

A user may choose to sync this information across all their devices.

The Screen Time setting must be specially turned on to be able to see data.

Screen Time – macOS 10.15+, iOS 13+ com.apple.ScreenTimeAgent, com.apple.remotemanagementd

Directories:

- macOS: /var/folders/<*darwin_user_dir*>/0/com.apple.ScreenTimeAgent/
- iOS: /private/var/mobile/Library/Application Support/com.apple.remotemanagementd/

Databases

- RMAdminStore-Cloud.sqlite
- RMAdminStore-Local.sqlite

Organized by Hour – Timed and Counted Items

- Usage by Category
- Application Usage via Bundle ID
- Web Browsing Usage
- Notifications
- Pickups

Data Retention – ~3 (iOS), ~5 (macOS) weeks

macOS: `/var/folders/<darwin_user_dir>/0/com.apple.ScreenTimeAgent/`
iOS: `/private/var/mobile/Library/Application Support/com.apple.remotemanagementd/`

Screen Time stores this information in RMAdminStore-*.sqlite databases. The data is organized by categories, hours, timed items, and counted items. This database will show general application usage, not granular application usage. This type of granularity will be discussed in the knowledgeC.db section.

References:
https://github.com/mac4n6/APOLLO/blob/master/modules/screentime_by_category.txt
https://github.com/mac4n6/APOLLO/blob/master/modules/screentime_by_hour.txt
https://github.com/mac4n6/APOLLO/blob/master/modules/screentime_counted_items.txt
https://github.com/mac4n6/APOLLO/blob/master/modules/screentime_timed_items.txt

Screen Time – Categories

Null entries are combined time between devices.

	HOUR	CATEGORY ID	CATEGORY TIME (MINUTES)	NAME	DEVICE ID	PLATFORM	GIVEN NAME	DSID
399	2021-05-26 17:00:00	Utilities	60.0	MBP-M1	418D1274-DE3A-56F9-86A6-EFD207EEA709	macOS	Elwood	1989733445
400	2021-05-26 18:00:00	Utilities	60.0	NULL	NULL	NULL	Elwood	1989733445
401	2021-05-26 18:00:00	Utilities	60.0	MBP-M1	418D1274-DE3A-56F9-86A6-EFD207EEA709	macOS	Elwood	1989733445
402	2021-05-26 19:00:00	Utilities	60.0	NULL	NULL	NULL	Elwood	1989733445
403	2021-05-26 19:00:00	Utilities	60.0	MBP-M1	418D1274-DE3A-56F9-86A6-EFD207EEA709	macOS	Elwood	1989733445
404	2021-05-26 20:00:00	Unspecified2	0.733333333333333	Elwood's iPhone	98a78b615f5b405d55c2023236a76e49cb75cf52	iOS	Elwood	1989733445
405	2021-05-26 20:00:00	Utilities	60.0	MBP-M1	418D1274-DE3A-56F9-86A6-EFD207EEA709	macOS	Elwood	1989733445
406	2021-05-26 20:00:00	Utilities	60.0	NULL		NULL	Elwood	1989733445
407	2021-05-26 20:00:00	Creativity	6.61666666666667	NULL		NULL	Elwood	1989733445
408	2021-05-26 20:00:00	Unspecified2	0.733333333333333	NULL		NULL	Elwood	1989733445
409	2021-05-26 20:00:00	Creativity	6.61666666666667	Elwood's iPhone	98a78b615f5b405d55c2023236a76e49cb75cf52	iOS	Elwood	1989733445
410	2021-05-26 21:00:00	Utilities	44.8666666666667	MBP-M1	418D1274-DE3A-56F9-86A6-EFD207EEA709	macOS	Elwood	1989733445
411	2021-05-26 21:00:00	Utilities	44.8666666666667	NULL		NULL	Elwood	1989733445
412	2021-05-29 11:00:00	Creativity	0.05	NULL		NULL	Elwood	1989733445

macOS: `/var/folders/<darwin_user_dir>/0/com.apple.ScreenTimeAgent/`
iOS: `/private/var/mobile/Library/Application Support/com.apple.remotemanagementd/`

One example of the data kept in this database includes usage by categories (Utilities, Social, Creative, Shopping, etc.). The query (on the next slide) output above matches the usage information shown in the screenshot on the previous slide.

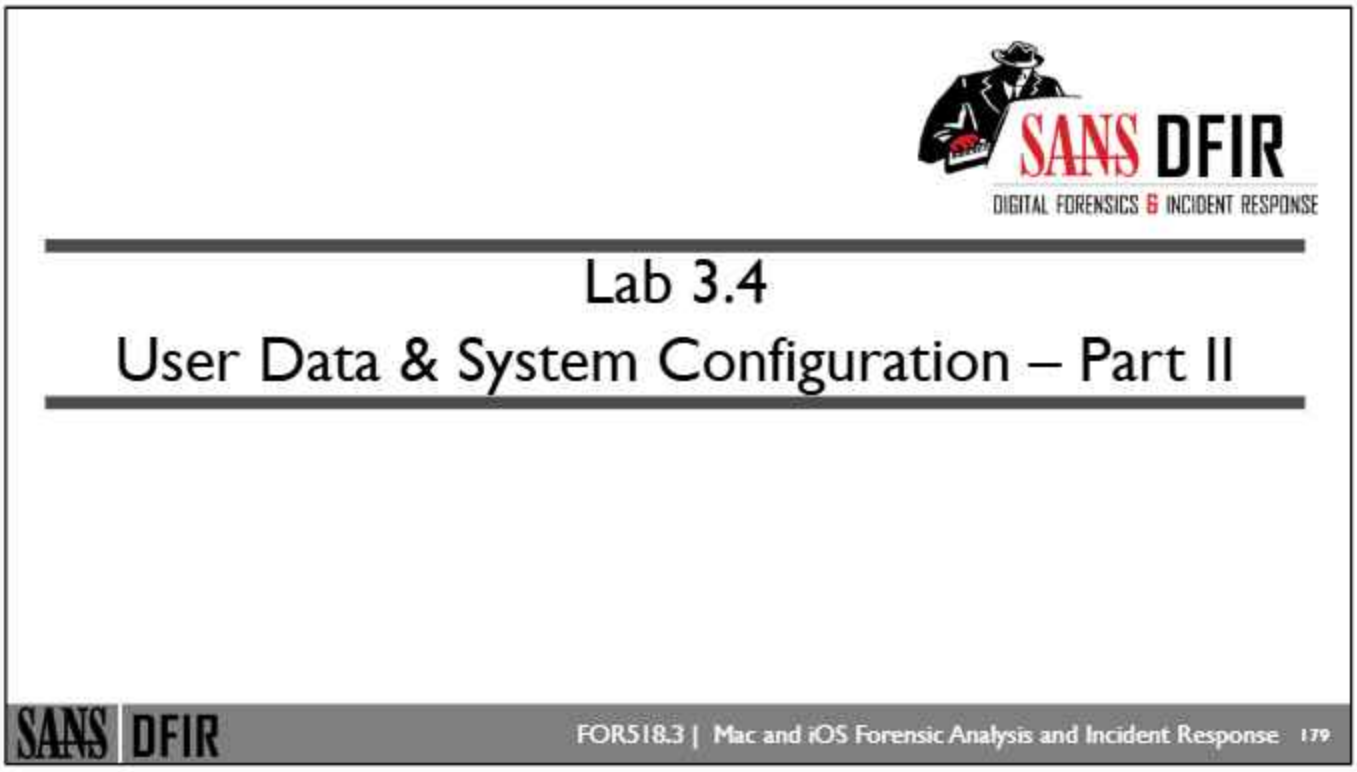
If the user chooses to sync the data across multiple devices, the name and UDIDs for those devices are shown. Some categories have not been specified by Apple and have been named as such. (This is usually when system applications such as phone, wallet, or camera apps are being used.).

The usage times of each device are shown as well as a combined time (for that hour) where the name and device columns are set to NULL.

References:
https://github.com/mac4n6/APOLLO/blob/master/modules/screentime_by_category.txt
https://github.com/mac4n6/APOLLO/blob/master/modules/screentime_by_hour.txt
https://github.com/mac4n6/APOLLO/blob/master/modules/screentime_counted_items.txt
https://github.com/mac4n6/APOLLO/blob/master/modules/screentime_timed_items.txt

```
SELECT
      DISTINCT
      DATETIME(ZUSAGEBLOCK.ZSTARTDATE+978307200,'UNIXEPOCH','LOCALTIME') AS 'HOUR',
      CASE ZUSAGECATEGORY.ZIDENTIFIER
         WHEN 'DH0011' THEN 'Unspecified1'
         WHEN 'DH0012' THEN 'Unspecified2'
         WHEN 'DH0013' THEN 'Unspecified3'
         WHEN 'DH1001' THEN 'Games'
         WHEN 'DH1002' THEN 'Social'
         WHEN 'DH1003' THEN 'Entertainment'
         WHEN 'DH1004' THEN 'Creativity'
         WHEN 'DH1005' THEN 'Productivity & Finance'
         WHEN 'DH1006' THEN 'Education'
         WHEN 'DH1007' THEN 'Information & Reading'
         WHEN 'DH1008' THEN 'Health & Fitness'
         WHEN 'DH1009' THEN 'Other'
         WHEN 'DH1010' THEN 'Unspecified4'
         WHEN 'DH1011' THEN 'Utilities'
         WHEN 'DH1012' THEN 'Shopping and Food'
         WHEN 'DH1013' THEN 'Travel'
         ELSE ZUSAGECATEGORY.ZIDENTIFIER
      END AS 'CATEGORY ID',
      ZUSAGECATEGORY.ZTOTALTIMEINSECONDS/60.00 AS 'CATEGORY TIME (MINUTES)',
      ZCOREDEVICE.ZNAME AS 'NAME',
      ZCOREDEVICE.ZIDENTIFIER AS 'DEVICE ID',
      CASE ZCOREDEVICE.ZPLATFORM
         WHEN 0 THEN 'Unknown'
         WHEN 1 THEN 'macOS'
         WHEN 2 THEN 'iOS'
         WHEN 4 THEN 'Apple Watch'
         ELSE ZPLATFORM
      END AS PLATFORM,
      ZCOREUSER.ZGIVENNAME AS 'GIVEN NAME',
      ZCOREUSER.ZDSID AS 'DSID'
   FROM ZUSAGETIMEDITEM
   LEFT JOIN ZUSAGECATEGORY ON ZUSAGECATEGORY.Z_PK == ZUSAGETIMEDITEM.ZCATEGORY
   LEFT JOIN ZUSAGEBLOCK ON ZUSAGECATEGORY.ZBLOCK == ZUSAGEBLOCK.Z_PK
   LEFT JOIN ZUSAGE ON ZUSAGEBLOCK.ZUSAGE == ZUSAGE.Z_PK
   LEFT JOIN ZCOREUSER ON ZUSAGE.ZUSER == ZCOREUSER.Z_PK
   LEFT JOIN ZCOREDEVICE ON ZUSAGE.ZDEVICE == ZCOREDEVICE.Z_PK
```

	HOUR	CATEGORY ID	CATEGORY TIME (MINUTES)	NAME	DEVICE ID	PLATFORM	GIVEN NAME	DSID
399	2021-05-26 17:00:00	Utilities	60.0	MBP-M1	418D1274-DE3A-56F9-B6A6-EFD207EEA709	macOS	Elwood	1989733445
400	2021-05-26 18:00:00	Utilities	60.0	NULL	NULL	NULL	Elwood	1989733445
401	2021-05-26 18:00:00	Utilities	60.0	MBP-M1	418D1274-DE3A-56F9-B6A6-EFD207EEA709	macOS	Elwood	1989733445
402	2021-05-26 19:00:00	Utilities	60.0	NULL	NULL	NULL	Elwood	1989733445
403	2021-05-26 19:00:00	Utilities	60.0	MBP-M1	418D1274-DE3A-56F9-B6A6-EFD207EEA709	macOS	Elwood	1989733445
404	2021-05-26 20:00:00	Unspecified2	0.733333333333333	Elwood's iPhone	98a78b615f5b405d55c2023236a76e49cb75cf52	iOS	Elwood	1989733445
405	2021-05-26 20:00:00	Utilities	60.0	MBP-M1	418D1274-DE3A-56F9-B6A6-EFD207EEA709	macOS	Elwood	1989733445
406	2021-05-26 20:00:00	Utilities	60.0	NULL	NULL	NULL	Elwood	1989733445
407	2021-05-26 20:00:00	Creativity	6.61666666666667	NULL	NULL	NULL	Elwood	1989733445
408	2021-05-26 20:00:00	Unspecified2	0.733333333333333	NULL	NULL	NULL	Elwood	1989733445
409	2021-05-26 20:00:00	Creativity	6.61666656666667	Elwood's iPhone	98a78b615f5b405d55c2023236a76e49cb75cf52	iOS	Elwood	1989733445
410	2021-05-26 21:00:00	Utilities	44.8666666666667	MBP-M1	418D1274-DE3A-56F9-B6A6-EFD207EEA709	macOS	Elwood	1989733445
411	2021-05-26 21:00:00	Utilities	44.8666666666667	NULL	NULL	NULL	Elwood	1989733445
412	2021-05-29 11:00:00	Creativity	0.05	NULL	NULL	NULL	Elwood	1989733445



This page intentionally left blank.