Global Information Assurance Certification Paper

Global Information Assurance Certification Paper

This is a GIAC Gold Template

Forensic Analysis on iOS Devices

GIAC (GCFE) Gold Certification

Author: Tim Proffitt, tim@timproffitt.com
Advisor: Hamed Khiabani

Accepted: November 5th 2012

Abstract

With a “bring your own device” (BOYD) movement, smart phones and tablets have exploded onto the corporate environment and show no sign of receding. This “consumerization” of endpoints means users will be performing work on devices other than the traditional organizational desktop or laptop running windows. Since smart phones and tablets are outfitted with more hardware than ever before they are being used to surf the internet, transfer data and to communicate with corporate mail servers. A large section of these BOYD devices are running Apple’s iOS and the ability to perform accurate and clear forensics on these devices will be valuable to an organization. This paper will cover the forensically sound methods that can be performed on an iOS device.

Introduction

Technology in smart phones and tablets is advancing in a feverish pace. Each release by the manufactures seems to embrace newer and more innovative technologies with ever expanding digital storage. Email, productivity suites, tasks lists, calendaring, browsing and presenting have all become common place on this platform. Many organization's workforces would be able to do large portions of their daily tasks on a tablet or mobile device if requested. With the design of the Apple Operation System (iOS) and the large amount of storage space available, records of emails, text messages, browsing history, chat, map searching, and more are all being kept. With the amount of information available to forensic analysts on iOS, this paper will cover the basics to accurately retrieve evidence from this platform and build forensically sound images when applicable. Once the image logically, via backup or physically has been obtained, files of interest will be highlighted for a forensic examiner to review.

The iOS Storage with HFS+ File System

The local storage on an iOS mobile device has several differences from the traditional Microsoft Windows or UNIX flavored workstation. Understanding these differences can help the investigator understand which tools to utilize and which actions to take when the results are not returned or what was returned was not expected.

In the early 90s Apple brought about a new file system. The Hierarchical File System (HFS) was designed to be a new dynamic file system and is formatted with a 512 byte block scheme to meet several new objectives by Apple. There are two types of blocks in the HFS system: logical blocks and allocation blocks. The logical blocks are numbered from the first block to the last block available on the volume and will remain static. Allocated blocks on the other hand are a bit different and can be tied together as groups to be utilized more efficiently by HFS. The structures of this file system include a volume header, startup file, allocation file, attributes file, extents overflow file and a catalog file (Morrissey 2010).

HFS+ Volume Header

Sectors 0 and 1 of the volume are the boot blocks. The volume header is utilized to contain information about the structure of the HFS volume. The header is the 1024 bytes after the reserved set of boot blocks on the partition. A backup of the volume header

exists and can be found in the last 1024 bytes of the volume. This backup is primarily used for disk repair if the original header is damaged or missing but is rarely used.

The Volume Header stores a wide variety of data about the volume itself, for example the size of allocation blocks, a timestamp that indicates when the volume was created or the location of other volume structures such as the Catalog File or Extent Overflow File.

HFS+ Allocation File

The purpose of the allocation file is to track which allocation blocks are used by the system or are free. The file specifies whether an allocation block is free by storing this data in a bitmap, specifying a free allocation block with a "clear bit". Zero means the block is free. The allocation file can also change size and does not have to be stored contiguously within a volume.

HFS+ Extents Overflow File

The extent overflow file tracks all allocation blocks that belongs to a file. The information recorded lists all extents used by a file and its' allocated blocks in the proper order. This information is stored in a balanced tree format.

HFS+ Catalog File

The catalog file describes the folder and file hierarchy on a volume. The catalog file contains metadata about all the files and folders on a volume including information on modified, access, and created times (Craiger, 2005). HFS uses a balanced tree catalog to allocate files. This catalog utilizes nodes to reference folders and files. The catalog file maintains the hierarchy of header, index, leaf and map nodes. The nodes are grouped together in a linear fashion to add speed to the process. Each file created is assigned a catalog ID number. HFS will increment the ID by one for each file added.

Partitions

An iOS device will have two partitions. The first partition is the firmware partition. The partition is a read only partition unless a firmware update is being performed. When an upgrade is performed, this partition is overwritten by iTunes with the new partition. This partition is normally between .9 and 2.7GB (depending on the size of the NAND drive) and will not have user data. This partition should be considered containing only system files, upgrade files and basic applications.

The second partition will contain user data. This partition will be the focus of

most investigations. This partition is where all iTunes applications will reside along with the user's profile data.

SQL Lite Databases

The SQLite data format is a popular format for mobile devices and open source applications. This database is relational and can be completely contained in a small C programming library. The SQLite database implements most of the SQL-92 standard but is missing some features. The format for this database is compact and contains some nice functionality for its size (http://www.sqlite.org/about.html).

Because of these features the iOS development community has embraced SQLite. Many of the native iOS applications such as Calendar, Text Messages, Notes, Photos, and Address Book utilize this database structure to store and organize their data. To be able to open and view this valuable evidence, a forensic examiner will need a stable database viewer. Sourceforge.net has a popular SQLite Browser that can be used to view most every iOS SQLite datastore (http://sourceforge.net/projects/sqlitebrowser/) or you can purchase RazorSQL (http://razorsql.com) for an inexpensive solution for under $100. For Firefox users there is a free SQLite Manager plugin.

Plists

The Property List (plist) is data file (sometimes called a property file) used to store various types of data on iOS and Macintosh operating systems. Originally Apple used the NeXSTEP format or a binary format for these files, but this was deprecated and a new XLM format was introduced. The examiner today will typically see either a XML or binary format.

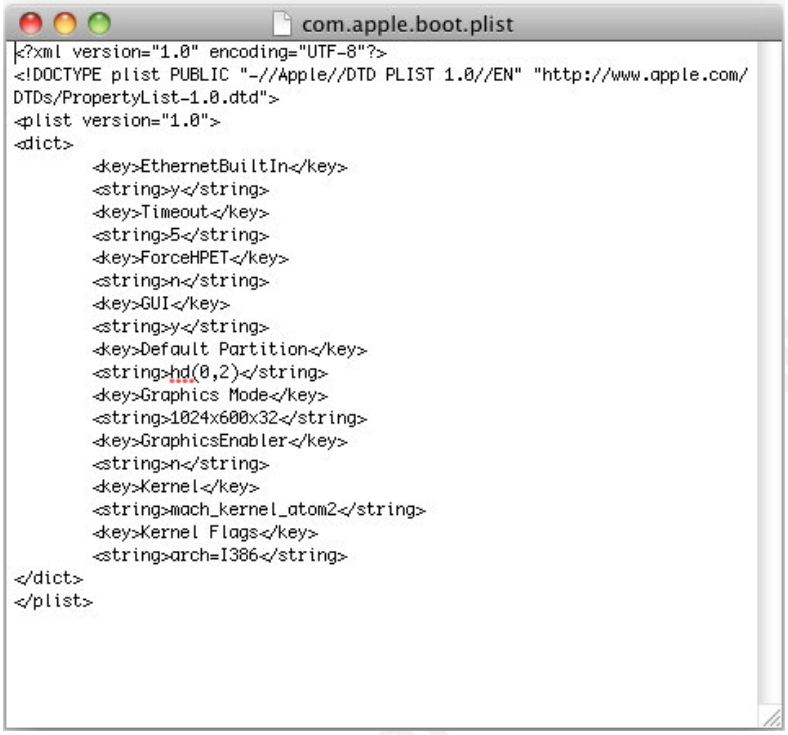

Figure 1

The plist in an iOS device can be used to store strings, dates, Boolean values, numbers or binary values (Hoog, 2011). Applications that maintain configuration data such as browsing history, favorites, configuration data, and others can place their data in a plist. When the examiner encounters a plist file that cannot be opened with a standard text editor a viewer will be needed. Plutil is a command line tool for Windows/Mac/Linux that can convert the binary plist files into human readable form for examination.

When a plist converts object into a XML property list it wraps the plist using tags (Apple, 2003).

Core Foundation Types with XML Equivalents

CF type	XML tag
CFString	<string>
CFNumber	<real> or <integer>
CFDate	<date>
CFBoolean	<true/> or <false/>
CFData	<data>
CFArray	<array>
CFDictionary	<dict>

Acquisitions

There are 4 major categories for acquiring forensics data from an iOS device. Each will have its positives and negatives and a forensic examiner may find he is utilizing several during an investigation. In some investigations you may not have the iOS device

in others you may have a locked and encrypted device. Regardless, the examiner should be aware of each of the 4 major categories and the limitations of each.

Passcode locked device are being utilized more frequently due to heightened security policies from organizations and general user awareness of theft. Circumventing the passcode is not always possible. The forensic examiner should first and foremost try to secure the passcode from the owner and/or immediately disable the passcode requirement if the phone is accessible. Setting the Auto-lock feature to "never" would be a desired setting for the duration of the investigation. The examiner should note that the setting was changed prior to taking a forensic image. A second setting to consider is to place the phone in "airplane mode". This setting would remove the ability for an outside entity to perform a remote wipe of the device thus tampering with the evidence after seizure.

The most common acquisition techniques include pulling data from an iTunes backup, pulling data from a logical API type method, jail breaking, and via obtaining a physical image of the storage hardware.

Acquisition via iTunes Backup

A popular approach, and one that is required when the iOS device is not available, is to analyze the latest backup of the iOS device. A backup would be retrieved from the workstation (Windows or MacOS) the device typically connects to for updates or syncing music. movies and applications. It should be noted that iTunes performs an automated backup during the sync process and/or when an upgrade to the iOS is performed. This configuration can be altered by the user. The backup(s) will be stored in alternate locations depending on the OS.

Windows XP	%systempartition%\documents and settings\ %username%\Application Data\Apple Computer\MobileSync\Backup
Windows 7	%systempartition%\Users\%username%\AppData\Roaming\Apple Computer\MobileSync\Backup\
MacOS	Users/%username%/Library/Application Support/MobileSync/Backup

Inside the backup folder there are several interesting files that will provide information about the device to be sure the examiner is reviewing the correct iOS device. The root of the backup folder will contain the status, info and manifest plist files. The Status.plist provides data about the latest backup. The Info.plist file contains data that can be used to confirm the backup matches the device. The IMEI number can be found here along with the phone number. See figure2.

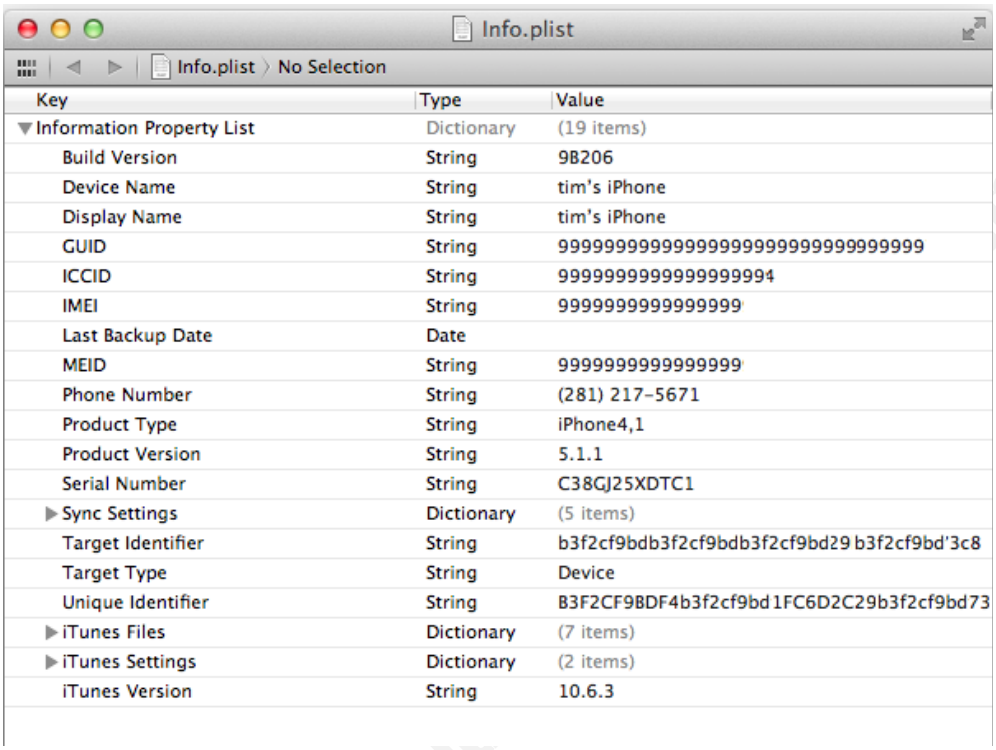

Figure 2

The Manifest.plist file contains metadata about the backed up files.

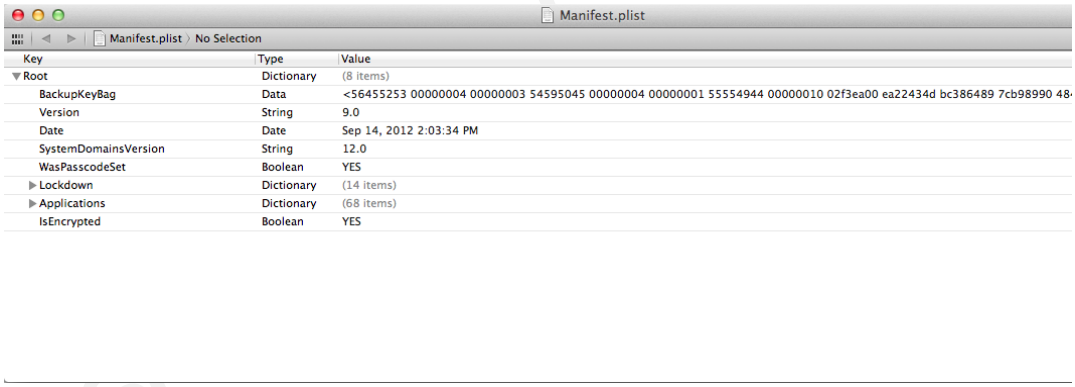

Figure 3

Author Name, email@addresstimproffitt.com

The backed up files themselves which are binary in nature are converted to a SHA1 hash value of the original filename. See figure 4. To view these files they must be converted to a legible, human readable format.

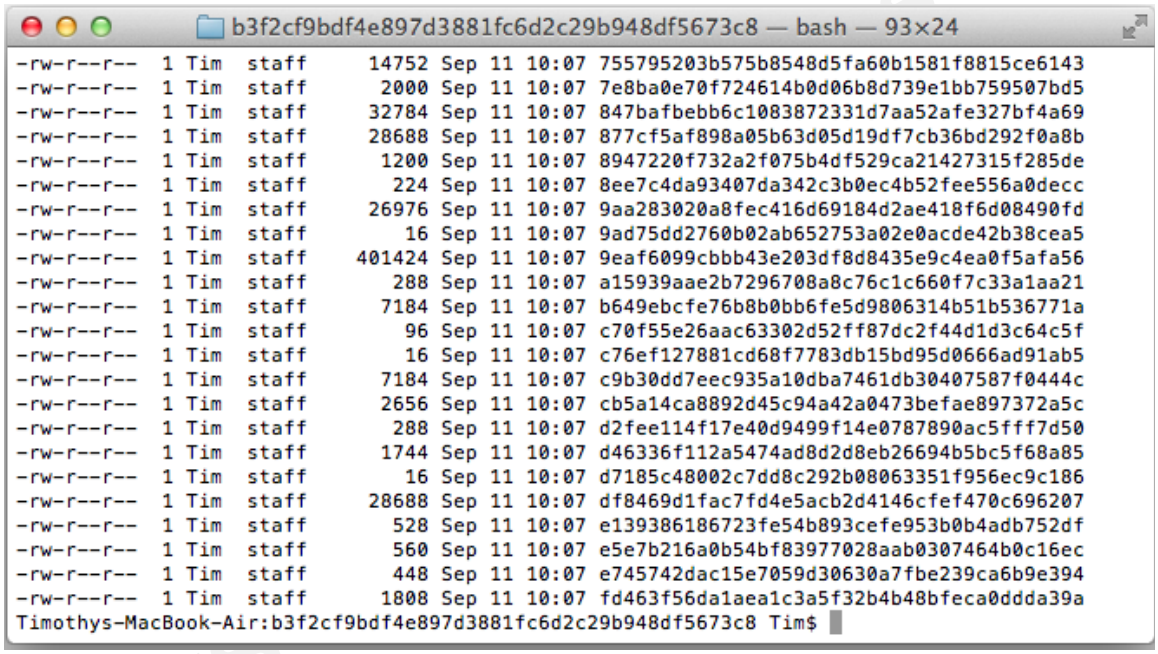

Figure 4

The *.mddata and *.mdinfo files are the binary files that contain the user data and will be the most interesting. Popular tools for review of this backup data include iPhone Analyzer, Paraben Device Seizure, iPhone Backup Extractor and Mobile Sync Browser.

1.1.1. Acquiring Backup Data with iPhone Analyzer

The iPhoneAnalyzer by Crypticbit is a free, java based, multi-platform tool designed to obtain data from an iOS backup. The iPhoneAnalyzer provides access to the file system from the iOS and a simple viewer into files that the forensic examiner may choose to preview. Because this is executed against the iTunes backup, the evidence is forensically sound due to no changes made to the data. The typical mode is to run the tool against a workstation backup from iTunes but also has the feature to create a backup directly from the phone if a previous backup is not available. An iPhoneAnalyzer feature

of note is “export all files”. This feature will convert the binary files of the backup to their proper names and locations for review on the forensic workstation. See figure 5.

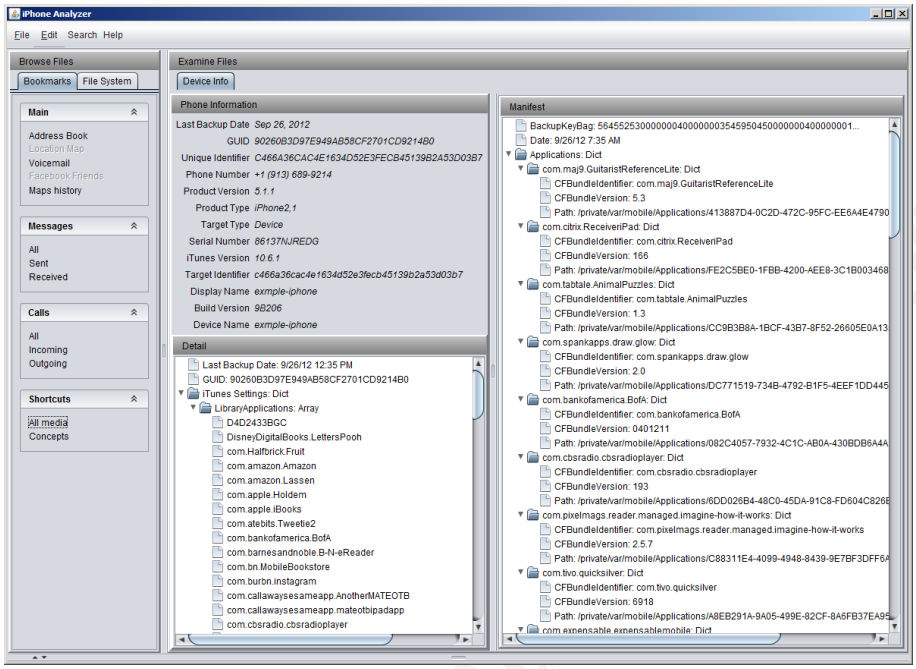

Figure 5

1.1.2. Issues with Encrypted Backups

iTunes offers the ability to encrypt backups which will prohibit an examiner from easily viewing the logical data. To be able to examine the backup data the examiner will need to run a password cracking utility against manifest.plist. This paper will not cover this topic, but there are tools for performing this task. Elcomsoft’s iPhone Password Breaker can be purchased for under $100 and can provide the password and keychain files as needed.

Alternately, jail breaking the phone (see section 5.4) can also bypass certain forms of a passcode by replacing the configuration files.

Acquisition via Logical Methods

The most popular approach today and one that has a growing market of tools sets is the logical acquisition method. Using this approach the allocated, active files on the iOS device are recovered and analyzed using a synchronization method built into the iOS operating system. This will allow the analyst to gather evidence on SMS, call logs, calendar events, contacts, photos, web history and email accounts. It must be understood

that this method will not allow for access to data in slack space. If evidence is suspected to be in slack space a physical acquisition is necessary. To acquire evidence using this method the examiner would cable the iOS device to their forensics workstation, run the software of choice and review the files of choice.

1.1.3. iPhone Explorer

The iPhone Explorer application developed by Macroplant offers a multi-platform Windows and MacOS installation. This product will allow the forensic examiner to quickly export data on call history, SMS, photos, contacts, bookmarks, etc. See figure 6. Some feature of this application will want the user to perform a backup so that the software can analyze the backup dataset. In each of these logical sections the iPhone Explorer will present date modified and if applicable size of the file. Using this tool, it is possible to obtain some data even after a factory reset of the device. Figure 7 details four calls that were listed after a "Reset All" was executed on the iOS device. Access to some data will be obscured depending on the iOS platform. Additional testing shows that an iPhone4s will not reveal call history, calendar, notes, contacts or messages due to the introduction of data protection techniques (Security Learn, 2012). A 3GS will show all evidence because the tool set can reach the files in the clear.

Figure 6

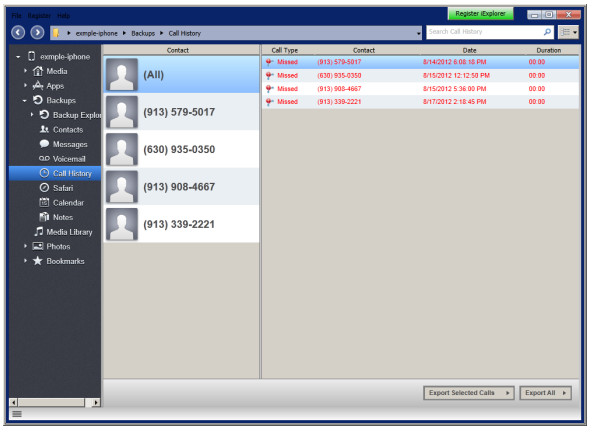

Figure 7

Acquisition via Physical Methods

The best case scenario for any forensic examiner is to obtain a bit by bit copy of the original media. The next sequential task would be to prove the copy and the original are exact. While forensically sound procedures are readily available and vetted for laptops and desktops, mobile devices such as iOS devices do not share the same luxury.
Since a physical acquisition method has the greatest potential for recovering artifacts of

the presented methods, researchers are continually attempting new techniques for achieving this goal on iOS devices. Since the storage medium on an iOS device is embedded, a set of challenges must be overcome. Since the examiner cannot remove the drive and connect it to his forensics workstation, specially crafted forensic software must be used in conjunction with the iOS devices design. A technique for acquiring an iPhone 2 image does not necessarily suffice for acquiring an iPad3 image. iOS version 3 can have different security methods than iOS version 6. Unfortunately, changing security models on the iOS device can keep an examiner from extracting a forensically sound image until a new technique is developed to grant privileged access.

Once physical image of the device is obtained it would allow the examiner to view additional items such as deleted items in unallocated space. There are a few organizations dealing in the Law Enforcement (LE) space that have tool developed for this. One of the original methods for obtaining an iOS acquisition was developed by Zdziarski (www.zdziarski.com). This method, now only available to LE, uses a technique of replacing the RAM disk software with a version that allows for the running of a live recovery agent capable of extracting the disk image. If the examiner is not a LE there are still a few options in the products Lantern and iXAM. Each of these toolsets modifies the RAM in order to execute forensic recovery agents on the operating system volume.

1.1.4. Lantern 2

The Lantern forensics suite developed by Katana Forensics INC (http://katanaforensics.com/) was designed to physically extract an image of the iOS device. At the writing of this paper this tool could extract or image data from any version iOS device running any software version.

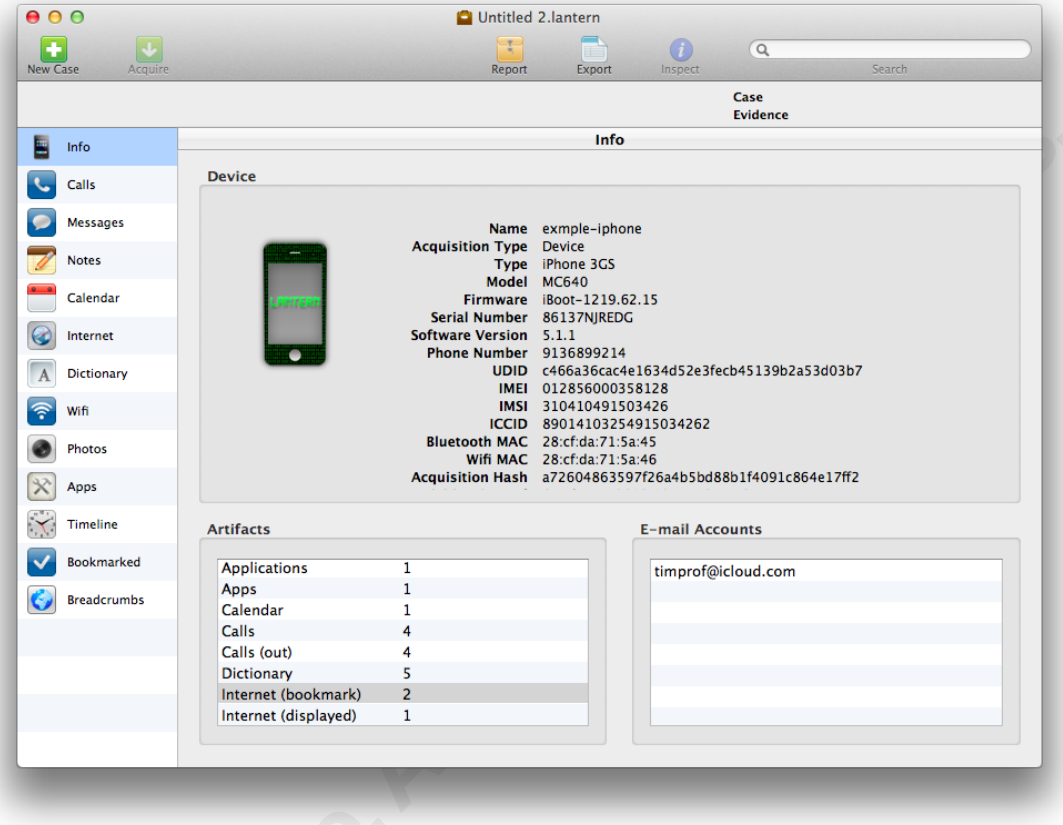

Figure 8

Lantern 2 (see figure 8) will quickly allow the examiner to review the most common pieces of evidence in a simple GUI interface. Plists and SQLite files will automatically be decoded and displayed for viewing. Lantern Imager is a complimentary application to Lantern 2 that was specifically designed to image iOS devices. Lantern imager can both decrypt the image and brute force a simple passcode (4 digits) along with providing a SHA-1 hash value.

1.1.5. iXam

iXAM (http://www.ixam-forensics.com/) pronounced *ig'zam* is designed to deliver evidence to a law enforcement investigation, providing anything from a stored contact or text message to an email, photograph or specific map location. The forensics' tool read is a byte level physical data copy which can be set to target specific data sets or the entire file system. iXAM does not modify the NAND flash and does not apply kernel patches used in jail breaking techniques. When used in forensic imaging mode, the output from iXAM is a raw disk image file in Apple's proprietary DMG format.

Acquisition via Jail Breaking

Jail breaking a phone is a technique used for replacing the firmware partition with a hacked version that will allow the examiner to install tools that would not normally be on the device (Three years of pwnage, 2012). With a functioning jailbreak on the iOS device the examiner will have tools not normally available such as SSH and Terminal. To obtain an image of the partition the iOS device must be jail broken. By far the most popular method for jail breaking is with redSn0w. The redSn0w tool has a simple wizard that will step the iOS device through the process of replacing the firmware and installing the Cydia application. Once the device has completed the process, the examiner can begin to extract artifacts.

To begin an extraction of the iOS device image, the forensic workstation would be placed on the same wireless network as the target iOS device. From the forensic workstation this SSH commands would start the process:

ssh root@172.16.103.106 dd if=/dev/rdisk0 bs=1M | dd of=ios-root.img

The SSH command from the forensics workstation connects to SSH server on the iOS device. The “dd if=/dev/rdisk0 bs=1M” executes the dd command with input file =/dev/rdisk0 and block size of 1M. The pipe then redirects the input to the next command dd of=ios-root.img which outputs the file ios-root.img onto the forensic workstation drive. The results will have an image file that can be manipulated by the forensic analyst’s choice of tools. It should be noted that this technique performed on an iPhone 3Gs and later will produce an encrypted image that cannot be parsed. For iOS devices utilizing hardware encryption of the user volume such as an iPhone4S, one of the physical acquisition tools mentioned above would be needed. These tools will decrypt the keychain needed to produce the readable image.

Analysis Tools

There are well known tools of the forensic trade that are capable of connecting to a mounded iOS image and providing analysis. Some of the open source community tools are extremely powerful and will allow the investigator to perform very specific searching and retrieval. Scalpel, DD, Find, Stings and others can be used on an iOS image much like that of a FAT or NTFS image. Major suites such as Encase and FTK imager can mount and analyze HFS+ images.

Relevant Evidence

An iOS device will have many Sqlite and plist files that can build a case for a forensic

examiner. The iOS operating system provides MACB (modified, accessed, changed, born date) times and can be vital when used with a timeline. Timelines are an essential element for forensic analysis and in the digital world and time stamps are recorded for many events (Eiland, 2006) listed in later sections. It should be recognized that many of the timestamps provided will be in CF Absolute Time, which means the number of seconds since Jan 1st, 2001 (Time Utilities Reference, 2010). To convert this use the formula =CreatedTime/(60*60*24)+DATE(2001,1,1). Alternatively a forensic examiner can use several online tools to translate this into a more human readable format such as www.epochconverter.com.

Files of Interest

The iOS directory structure is common across all iOS devices. The folder structure resembles a UNIX layout and there are several directories that the examiner will immediately be interested in. Some files will be stored in text format and easily readable. Other files will be stored in SQLite databases, XML files or binary. The default applications store their data in the private/var/mobile/Library folder. This includes the Address Book, Mail, Calendar, Maps, Notes, YouTube, Safari, Texting, Weather and Voicemail applications. Downloaded applications from iTunes such as NFL 2012, Shazam or AroundMe will store their data in private/var/mobile/Applications.

1.1.6. iTunes Applications in private/var/mobile/Applications

When an application is obtained from the iTunes store, a new directory is automatically created in the Mobile/Application folder. This directory will hold the files associated with each application and will be assigned a 32 character alphanumeric unique identifier by Apple (Example: GA07A3WW- 0E39-33OJ-B947-9CAA16688G22). This unique id will be consistent across all iOS devices. Each application folder will typically have several common subfolders:

documents folder for relevant files to that application
temp folder for temporary runtime files
library folder for preferences, cached data

Common files are found within most applications folders such as info.plist, resourcerules.plist and applestores.db. Depending on the application, varying configuration files, plist files and XML data will be found. The examiner can occasionally find username and password data, cookies, or images that will help provide evidence for

the investigation.

1.1.7. Photos in private/var/mobile/media/DCIM

On iOS devices, this folder will contain photos either taken or synced to the device. The pictures found here will have timestamp metadata. Photos within the 100APPLE folder indicate that they were taken from the device itself. The camera application numbers the images from the iOS device sequentially. The first picture taken will be named IMG_0001 and will continue numbering without regards to files being deleted or moved. This is an interesting fact for the forensic examiner in that if an IMG numbered image is missing it can be assumed it was deleted.

iOS devices also have the ability to take screenshots of itself. These images can give the examiner a view into what may have been installed on the machine prior to its current state. These files can be found in the DCIM/999Apple folder. If a contraband or corporately banned application was suspected to have been on the device, this would be a location to help prove this was the case.

1.1.8. Keystrokes in /private/var/mobile/Library/Keyboard

The dynamic dictionary is the text file dynamic-text.dat. This dictionary stores words typed by the user during the course of using the device. Any word entered into applications like Notes, Safari, Messages, Facebook or any application that will allow text input, will be captured. The intent of this file would be to aid the user in typing. Consequently this can help the forensic examiner highlight the user's common words or build a case for an interesting keyword for searching. Unfortunately there is no timestamp captured with this dictionary. Words found in the dictionary could have been typed at any point in the life of the device.

The UserDictionary.sqlite database contains a user's manual auto-corrections. This database can also contain interesting evidence of technical or special keywords that may not be Standard English words or acronyms that could be helpful for the investigation.

1.1.9. Passwords in /private/var/Keychains

Many of the iOS applications use Apple's keychain for password management. The key-chain-2.db file contains several tables (cert, genp, inet, keys, sqlite_sequence, and tversion) that are known to contain accounts and passwords that the device has used in the past. Voicemail passwords, wireless access point key phrases and device login passcodes can be found inside this database as well. In some cases the passwords will be encrypted by the iOS encryption keychain procedure and will need to be decrypted.

Obtaining Elcomsoft's iPhone Password Breaker, an examiner can provide this tool the extracted keychain file and decrypt these files.

1.1.10. Notes in /private/var/mobile/Library/Notes

The default notes application can be a treasure trove of keywords and nice evidence for an investigation. The notes.sqlite database contains 9 tables with ZNote being the most important. This table has CREATIONDATE (Epoc timestamp), MODIFICATIONDATE (Epoc timestamp), and ZTITLE which contains the title of the note. The table ZNOTEBODY will contain the contents of the note in the ZCONTENT column.

1.1.11. Text Messages in /private/var/mobile

SMS and text messages can be one of the most sought after pieces of evidence by the authority requesting the examination of the device. Most organizations have the ability to extract and read a user's mailbox but rarely have the tools or monitoring in place to review text messaging communication that is conducted on an iOS device. Inside /private/var/mobile/Library/SMS the sms.db can be found. The sms.db sqlite database can house both existing and deleted conversations. The database has 6 tables but the table "message" and "msq_pieces" will contain the majority of the interesting evidence.

The message table contains a row for each message. See figure 9. The column data for each record contains details including a rowID, a date (EPOX format), phone number, the message in the text, and whether the message was sent or received. The column "flags" indicates if the message was sent (3) or was received (2). The column "read" will hold the value of one if the message was read.

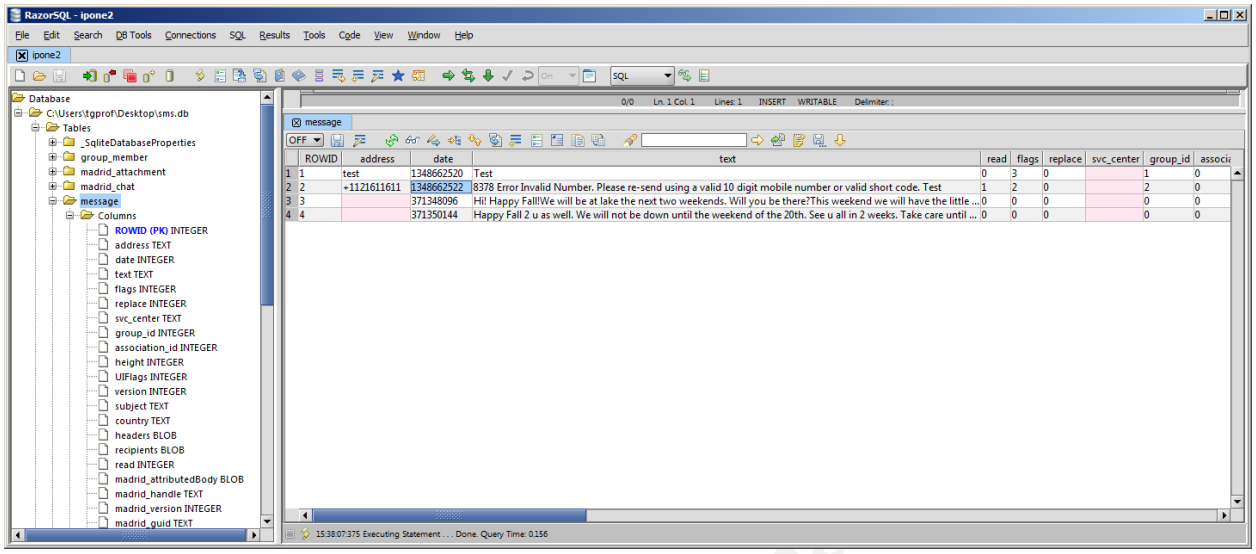

Figure 9

The msg_pieces table contains a row for each message that contains content that would be sent along with the message. In most cases this would be a photo or video. The column message_id, which will be assigned to each message, will correspond to the rowID of the message table. There can be three records corresponding to the original message with the third records holding the photo or video that was attached. The "data" column will contain the text content attached with the message.

1.1.12. Browser Cookies in /private/var/mobile/Library

Safari cookies can be an important piece of evidence when identifying web browsing from the device. iOS applications store their persistent cookies in the cookies.binarycookies file. This will be different from other browsers such as Internet Explorer in that this file is in a binary format as opposed to plain text files in the history folder or in a SQLite database like chrome. To read this file, the examiner would need to use a tool listed above like iPhone Extractor or open the file in a HEX editor. The file is composed of a header followed by one or more pages. Each page is comprised of one or more cookies (Miyake, 2011).

Field Name	Size	Description
Signature	4 bytes	"COOK" header
Number of pages	4 bytes	Little Endian Integer
Page Size	4 bytes	Little Endian Integer
Page	X bytes	Variable size data for the cookie.
Tail	8 bytes	Possible Hash for checksum.

1.1.13. Browser Searches in /private/var/mobile/Library/Caches/Safari

Search terms from using the Safari browser can be found in the RecentSearches.plist file. This particular plist file is in XML format and can be read with a text viewer.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
      <key>RecentSearches</key>
      <array>
            <string>corvette emergency brake line diagram
</string>
            <string>who won the debate</string>
      </array>
</dict>
</plist>
```

Figure 10

In the above example (figure 10) you can see the terms “who won the debate” and “corvette emergency brake line diagram” were used in the Safari search option.

1.1.14. AddressBook in /private/var/mobile/Library/AddressBook

The address book in the iOS can contain a wealth of information about the owner’s personal contacts. The AddessBook.sqlitedb file contains several tables of which two are of interest. The ABPerson table will contain first, last, organization, notes, birthday, job title, nickname, prefix and more. It appears that the index of this table is called ROWID. The ABMultiValue table will contain in the element “value” the email or phone number of the individual in the ABPerson table. These two tables are linked by a one to many relationship of the ABPerson(ROWID) and the ABMultiValue(record_id).

4.1.10 Call History in /private/var/Library/CallHistory

The call history of an iOS device that can place cellular calls is contained in call_history.db. This SQLite database has 4 tables. The call table in this database contains the phone number, date, duration and reference ID of the contact. The date field will need to be converted from EPOC time format. The id field in this table can reference the address book ID from the section above. When a negative one is listed there was no contact. The flags field in the calls table will indicate incoming or outgoing calls. Generally a number four indicates inbound call and a number 5 is an outbound call.

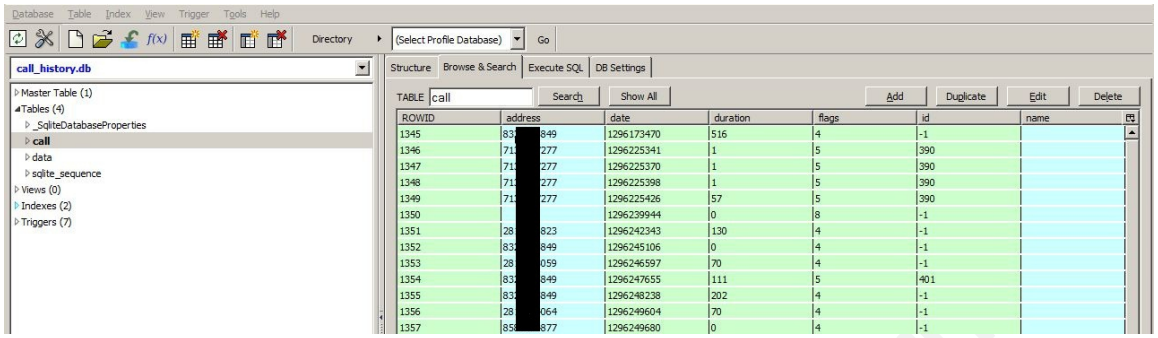

Figure 11

Geographical Location and Wi-Fi Data

GPS and Wi-Fi evidence can be a sought after item to help build a picture of the iOS device location at a specific time and also users habits. Many iOS applications will attempt to cache the user's location and store GPS data depending on the purpose of the application. A prime example is the iPhone's camera will attempt to store longitude and latitude when a photo is taken. The consolidated.db file found in private/var/Library/Caches/locationd can hold a tremendous amount of geolocation data. The database contains geolocation data for every cell tower the iOS devices utilizes. It has been designed to track both GPS and Wi-Fi data in one location for applications to utilize. This SQLite database has several tables that are of interest to the examiner.

Author Name, email@addresstimproffitt.com

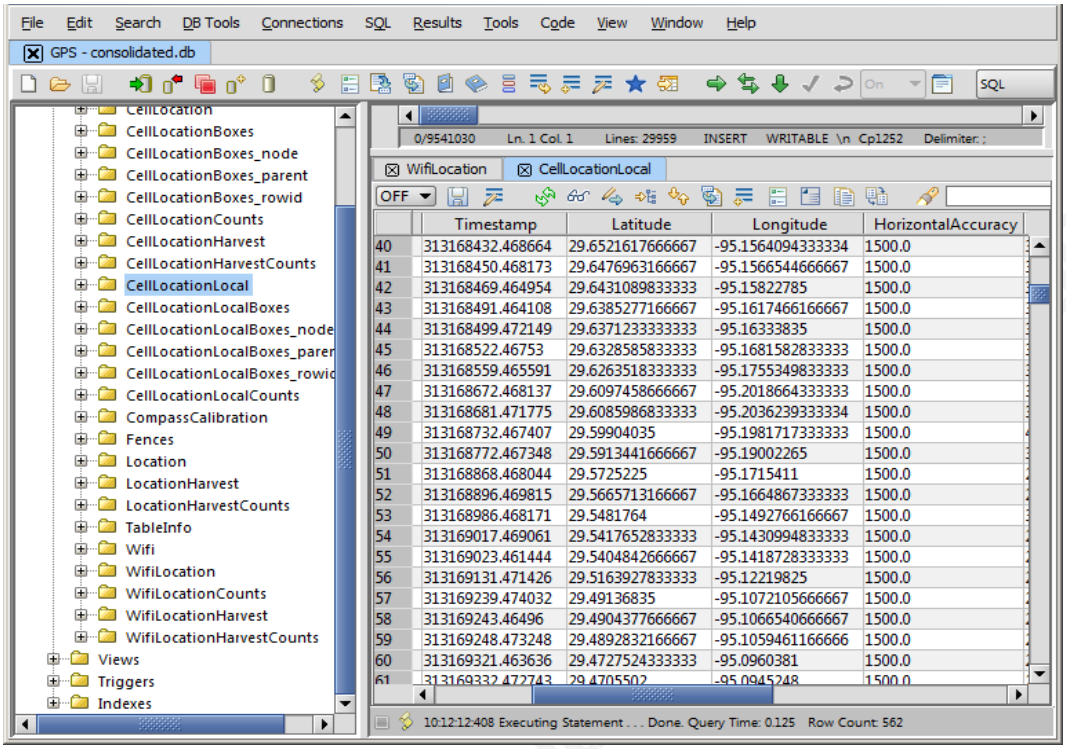

Figure 12

The wifilocation table contains longitude, latitude, MAC, and timestamps of wireless infrastructures the iOS device has utilized.

The cellLocationLocal table contains longitude, latitude, altitude, timestamps and tower data.

Timestamps in this database can be a bit confusing. The time is neither CF nor EPOC. iOS devices for this table use the number of seconds from 1/1/2001 to present. The formula would look like =(((TIMESTAMP/60)/60)/24)+DATE(2001,1,1)+(-6/24). Using a spreadsheet to perform the calculations and Google maps to input the longitude and latitude, and examiner can find geographically where a phone was at a point in time. If a more polished tool is requested, the iPhone Tracker from Peter Warden's website (peterwarden.github.com/iPhoneTracker) will pull the consolidated.db file from a backup and graphically display the locations the iOS device has been over time.

Conclusion

iOS devices collect and store a tremendous amount of evidence about a user's activities. In many cases one could argue more evidence is collected than the user may want. Locations, messages, contacts, web surfing habits, notes, pictures and more are available on iOS devices storage media, many with time stamped data. With this forensic

Author Name, email@addresstimproffitt.com

evidence available, and more business being conducted on iOS devices, forensic examiners need to be able to successfully and accurately pull this evidence when requested by authorized authority. By utilizing proven, existing forensic techniques along with specialty tools mentioned in this paper, examiners can collect and present evidence from an iOS device. This evidence can then produce a clear report of the activities performed on the device.

References

Morrissey, Sean. (2010) iOS Forensic Analysis: for iPhone, iPad and iPod Touch. New York, NY: Apress

Craiger, Phil, Burke, Paul. Mac Forensics Mac OS X and the HFS+ File System. Retrieved from http://www.google.com/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=2&cad=rja&ved=0CCYQFjAB&url=http%3A%2F%2Fwww2.tech.purdue.edu%2Fcit%2FCourses%2Fcit556%2Freadings%2FMacForensicsCraiger.pdf&ei=Z4KKULrNNaPg2AXBoYDACA&usg=AFQjCNHkVG3kp3_vOBt7wbIDz8UvPklKTg&sig2=xctGMJrfEjyH8kCntEittw

Mac Developer Library. (2003). BDS File Formats Manual. Retrieved from https://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man5/plist.5.html

Eiland, Earl. (September 2006). Time Line Analysis in Digital Forensics. Retrieved from http://www.google.com/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=4&cad=rja&ved=0CEcQFjAD&url=http%3A%2F%2Fwww.cs.nmt.edu%2F~df%2FStudentPapers%2FEiland.pdf&ei=91uBUPnYG6TC2QXuhIC4AQ&usg=AFQjCNGepnhtUCKE7ZVREvaO-YD6yy9_FQ&sig2=tbmKIGaRDIWdbBxX3_8DbQ

Hoog Andrew, Strzempka Katie. (2011). iPhone and iOS Forensics: Investigation, Analysis and Mobile Security for Apple iPhone, iPad and iOS devices. Waltham, MA. Syngress

Security Learn Blog. (January 2012). Retrieved from http://www.securitylearn.net/2012/01/10/iphone-forensics-on-ios-5/

Mac Developer Library. (2007). Time Utilities Reference. Retrieved from https://developer.apple.com/library/mac/#documentation/CoreFoundation/Reference/CFTimeUtils/Reference/reference.html

Miyake, E. Safari 5.1 Cookie.binarycookie File Format. Retrieved from

http://www.tengu-labs.com/documents/Miyake%20-%20Safari%20Cookie.binarycookie%20Format%200_2[Draft].pdf

Last Updated: April 8th, 2014

Upcoming Training

Mentor Session - FOR 408	**Mexico City, Mexico**	**May 03, 2014 - Jul 05, 2014**	**Mentor**
SANS Security West 2014	**San Diego, CA**	**May 08, 2014 - May 17, 2014**	**Live Event**
SANS Melbourne 2014	**Melbourne, Australia**	**May 12, 2014 - May 17, 2014**	**Live Event**
Community SANS Madrid FOR408 (in Spanish)	**Madrid, Spain**	**May 26, 2014 - May 31, 2014**	**Community SANS**
Digital Forensics & Incident Response Summit	**Austin, TX**	**Jun 03, 2014 - Jun 10, 2014**	**Live Event**
SANS Rocky Mountain 2014	**Denver, CO**	**Jun 09, 2014 - Jun 14, 2014**	**Live Event**
Mentor Session - FOR 408	**San Antonio, TX**	**Jun 19, 2014 - Aug 28, 2014**	**Mentor**
SANSFIRE 2014	**Baltimore, MD**	**Jun 21, 2014 - Jun 30, 2014**	**Live Event**
SANS vLive - FOR408: Computer Forensic Investigations - Windows In-Depth	**FOR408 - 201407,**	**Jul 15, 2014 - Aug 21, 2014**	**vLive**
Mentor Session - FOR 408	**Mexico City, Mexico**	**Jul 16, 2014 - Sep 17, 2014**	**Mentor**
SANS Boston 2014	**Boston, MA**	**Jul 28, 2014 - Aug 02, 2014**	**Live Event**
SANS Virginia Beach 2014	**Virginia Beach, VA**	**Aug 18, 2014 - Aug 29, 2014**	**Live Event**
SANS Crystal City 2014	**Crystal City, VA**	**Sep 08, 2014 - Sep 13, 2014**	**Live Event**
Community SANS Paris @ HSC - FOR408 (in French)	**Paris, France**	**Sep 15, 2014 - Sep 19, 2014**	**Community SANS**
SANS Albuquerque 2014	**Albuquerque, NM**	**Sep 15, 2014 - Sep 20, 2014**	**Live Event**
SANS Baltimore 2014	**Baltimore, MD**	**Sep 22, 2014 - Sep 27, 2014**	**Live Event**
SANS DFIR Prague 2014	**Prague, Czech Republic**	**Sep 29, 2014 - Oct 11, 2014**	**Live Event**
SANS Seattle 2014	**Seattle, WA**	**Sep 29, 2014 - Oct 04, 2014**	**Live Event**
SANS vLive - FOR408: Computer Forensic Investigations - Windows In-Depth	**FOR408 - 201410,**	**Oct 06, 2014 - Nov 12, 2014**	**vLive**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**