The Quarterly Magazine for Digital Forensics Practitioners

Issue 26 • February 2016

DIGITAL FORENSICS / MAGAZINE

WIN! an iPod Nano

iOS FORENSICS

Latest News, 360 Book Reviews, IRQ
& much more inside!

PLUS!

Steganography & Metadata
Improving Trust in the Cloud
Mobile Device Location
Malvertising

9 772042 061004 26
Issue 26 / £14.99 TR Media

iOS 9 FORENSICS

Mattia Epifani and Pasquale Stirparo provide an overview of the current state of the art in iOS acquisition methods with some insight into Logical Acquisition Analysis.

/ INTERMEDIATE

iOS 9, the new and latest version of Apple's mobile operating system, was released in September 2015 and it is supported starting from iPhone 4s and iPad 2. Its introduction has increased the level of complexity for the forensics practitioners who have to analyse Apple's mobile devices. Due to (or thanks to) security enhancements, there are several functionalities that make it harder, particularly the acquisition phase. In this regard, it is advisable to read Apple's "iOS Security Guide" [1], latest version released in September 2015.

/ DEVICE ACQUISITION

In general, in the field of mobile forensics we can distinguish between three main different types of acquisition: physical, file system, and logical.

Regarding the physical acquisition, at the moment there are no known vulnerabilities at bootrom level, such as those found in devices up to iPhone 4, which would allow a physical acquisition of the device (or a copy bit-by-bit of the internal memory), therefore this road is not generally feasible for devices with the iOS 9.

In the case of devices that have been jailbroken by the user, however, one can perform a physical acquisition of the device by accessing it directly or using the functionality of software like Elcomsoft iOS Forensic Toolkit. The full physical acquisition (bit-by-bit image) works on all 32-bit devices, while in the case of 64-bit devices it results in a complete file system acquisition.

/ Q&A

What if the backup is encrypted and I'm not able to recover/break the password?
In the case where the backup is encrypted and it has not been possible to obtain the password, the analysis is limited to data obtained through the protocol AFC:

- Images and videos with their metadata
- iTunes Library (folder iTunes_Control)
- iBook
- Files contained inside applications that make use of iTunes File Sharing

Over the years, different approaches have been proposed for the logical acquisition. The most popular ones are:

1. Backup made through iTunes
2. Access through the Lockdown service
3. Extraction through the AFC protocol

The second method, proposed by Zdziarski in 2014 [2], had already been rendered unusable with iOS 8, while the third method was severely limited starting from iOS 8.3.

For these reasons, at the moment the most commonly used practice is to create a backup of the device via iTunes or software that uses iTunes libraries (e.g. libimobiledevice [3]). However, the scenarios that an investigator may face are multiple and could affect the ability to obtain such backup or to be able to interpret the content.

> *"iOS 9, THE NEW AND LATEST VERSION OF APPLE'S MOBILE OPERATING SYSTEM, HAS BEEN RELEASED IN SEPTEMBER 2015 AND IT IS SUPPORTED STARTING FROM IPHONE 4S AND IPAD 2."*

The first problem that arises is the use of a device passcode by the user, which makes the phase of search and seizure of the device even more crucial as we are going to see in this article.

In general we can distinguish four cases:

1. The device is switched off at the time of the seizure and locked with a passcode
2. The device is switched on and unlocked at the time of the seizure, but it has a passcode set in place
3. The device is switched on and locked at the time of seizure
4. The device is either switched on or off at the time of the seizure and no passcode is in place.

In the first case there are no known techniques for bypassing the passcode and so it is not possible to access the data contained in the device. The only information that can be recovered is that about the device; more precisely Device class (iPhone, iPad, iPod), Device name, Device colour, hardware model, iOS version, Unique Device ID (UDID) and Wi-Fi MAC Address.

In the second case it is crucial to get a backup of the device as quickly as possible, and for this reason it is advisable to keep the device switched on, isolate it from possible connections with networks and other devices by placing it in Airplane mode, check if the device has a passcode set (Settings ⇢ General ⇢ Passcode) and deactivate the automatic locking of the device (Settings ⇢ General ⇢ Auto-Lock). All these operations must be carried out taking care not to click on the power button, which would lock the device and make the acquisition more complex or even impossible. At this point you need to connect the device to an acquisition box (e.g. a workstation with forensic software

installed or simply with iTunes) to enable the coupling of the device and the backup.

In the third case, which may also occur for an error in the management of the previous case, the only way we could get a backup is to look for a lockdown certificate on a computer previously paired with the device (e.g. a computer of the owner), while keeping the device switched on and power on. These certificates are easily identifiable because they are located within a specific folder, depending on the computer's operating system, and also because the file name corresponds with the UDID phone's identifier (which, remember, can be extracted even with the phone locked). In Microsoft operating systems since Windows 7, certificates are located under \ProgramData\Apple\Lockdown\, while in a Mac OS X are located under /var/db/lockdown/. Once a certificate has been found, it can be copied to the acquisition workstation in the same folder and used to get a backup of the target device.

Finally in the fourth case, the simplest for the investigator, it is always possible to pair the device with the acquisition box and obtain a backup.

/ WHY CAN CRASH LOGS BE USEFUL?

Crash Logs are used by the iOS operating system to provide developers with indicators relating to problems generated by their Apps. From a forensics point of view, they can be used to create a timeline of the events contained in the log and to understand the range of activities of both operating system and applications (e.g. information about the use of applications which have been already uninstalled from the device at the time of the acquisition). Some of the most interesting Crash Log files are more those generated by Apple's Mail app: within these log files are the names of attachments received via e-mail. This is information that is not possible to extract from non-jailbroken devices because the emails are not included in the backup and cannot be accessed with the protocol AFC. Moreover, you can also find the names of attachments of emails that have been subsequently deleted. For each attachment is the date of the email has been received, the email account of reference and, of course, the file name. Here is an example of how a crash log entry looks like:

2015-11-13 19:42:21.730|[145:0x156082e0]|LogAttachments: [Attachment] Unable to read file URL [file:///var/mobile/Library/Mail/IMAP-mattia.epifani@realitynet.it@imap.gmail.com/%5CInbox.imapmbox/Attachments/207559/2/ESAME_Forensics.docx]

Even when it is possible to extract data from the device via the backup acquisition, it is advisable also to make an acquisition of data that can be obtained directly from the device via the AFC protocol. This can be done via any forensic software (e.g. Oxygen Forensics or Magnet Acquire), or simply by using a management software that supports the AFC protocol (e.g. iFunBox). This choice is motivated by the fact that the user may have to set a password on the backup, making it very difficult if not impossible to access the data once extracted.
By acquiring the data via AFC protocol, in fact, the extracted data is in cleartext regardless of the presence of a password on the backup.

A further aspect not to be underestimated is the acquisition of Crash Logs of the device: the access to such data is permitted under the same conditions described before for backup accessing the device directly. If you have access to a workstation running Mac OS X you can use XCode, while under Windows you can use the software iTools (http://pro.itools.cn/itools3_en).

/ UNDERSTANDING THE BACKUP STRUCTURE

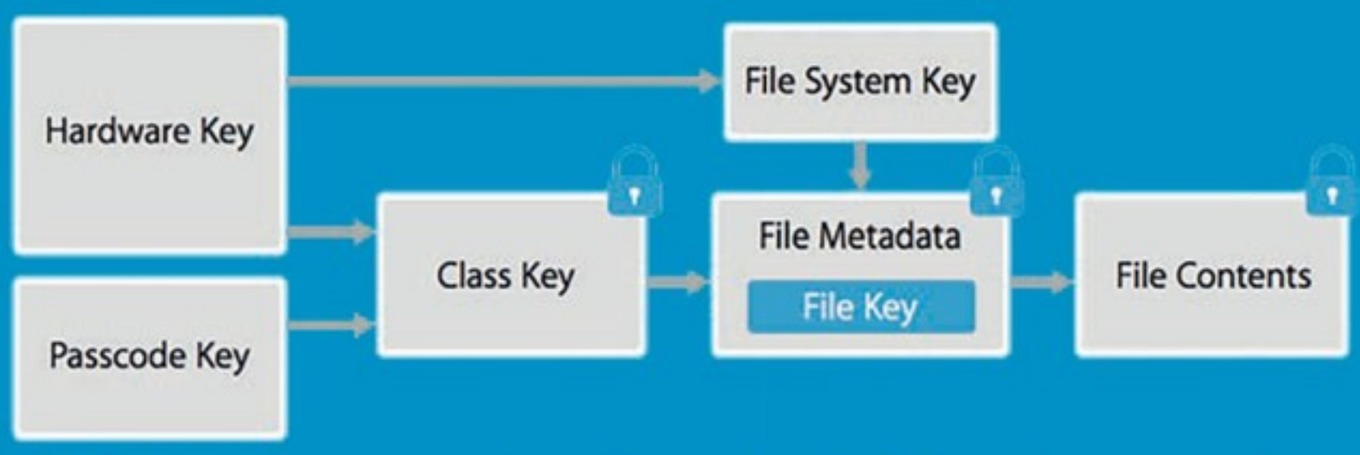
As you know, Apple allows the user to make a backup locally and/or using the iCloud service. If the user chooses to make a local backup, he/she can set a password that will be used to encrypt the backup. This password is stored within the device itself and used for all subsequent backups. For this reason the acquisition just made may actually be encrypted and its data not directly accessible.

/ WARNING

We have already mentioned this in the article, but it is quite a common mistake so it's important to underline it once again. Given how hard it became to bypass Apple's security mechanisms, one of which is the lock screen, it is fundamental for the investigator that gets an unlocked device to keep it as such. Most common causes for loosing an unlocked device we have seen are a) accidentally pressing the power button, b) forgetting to plug the power cable during the transportation to the lab and c) placing it unlocked but forgetting about the auto-lock option still active.

/ DATA CARVING IN IOS

Apple uses a technology called Data Protection in order to protect data stored in their iOS devices. For each new file created, a new 256-bit per-file key is generated and used to encrypt the file content using AES encryption. The per-file key is then wrapped with one of the data protection class keys and stored in the file's metadata, which is in turn encrypted with the file system key (the EMF key) that is generated from the unique hardware UID. The following image from Apple "iOS Security Guide" summarizes the process.



Clearly classic file carving procedure will not work, since in the unallocated space there will only be encrypted content. An interesting approach on how to possibly carve deleted images from iOS devices has been published by D'Orazio. He suggested comparing the catalog file and the journal file from HFS+ to identify information about deleted files. Based on this information the analyst should be able to search and recover deleted files, locate the cryptography keys, and then decrypt the image file. However, that approach may work only if the device has not been restored, wiped, or upgraded to a new iOS version, because in such cases a new file system key (EMF) would be recreated.

"AT THE MOMENT THERE ARE NO KNOWN VULNERABILITIES AT BOOTROM LEVEL, SUCH AS THOSE FOUND IN DEVICES UP TO IPHONE 4, WHICH WOULD ALLOW A PHYSICAL ACQUISITION OF THE DEVICE."

It is useful in any case to remember that the generation of a backup produces four files that are in each case in the cleartext:

- Info.plist
- Manifest.plist
- Status.plist
- Manifest.mbdb.

By analysing the file manifest.plist it can be easy to understand if the backup is encrypted or not (field IsEncrypted that can take values true/false). From the analysis of the plist files it is also possible to identify the following information: date of backup creation, phone number, device name, GUID, ICCID, IMEI, product type, IOS version, serial numbers, UDID, the iTunes software used to create the backup (iTunes version number and iTunes settings) and the list of applications installed on the device backed up.

Manifest.mbdb is a binary file that stores the descriptions of all the other files in the backup directory. It contains a record for each element in the backup (comprising symbolic links and directories, which of course do not have a corresponding element among the backup files). Each record contains the following parameters: Domain, Path, Link Target, User ID, Group ID, Modified Time, Access Time, Creation Time, File size, Unix file permissions, File hash. The first level of the hierarchy of the backup files is their domain. The domain for each file is written in its corresponding record in the Manifest.mbdb file. Each file has a domain name (e.g. Camera Roll Domain contains multimedia elements related to the Camera application such as images, videos, etc.).

If the extracted backup is unencrypted, we can then proceed with its interpretation, i.e. reading the content of the file manifest.mbdb in order to rebuild the structure of files and folders. If the acquisition has been done using a forensic software, the same will typically also have features for interpretation

of the files structure and content. Alternatively, if the acquisition has been carried out with iTunes, the backup can be imported in forensic software or interpreted with non-forensic specific software.

On the other hand, if the backup is encrypted it is possible to try the approach of the attack on the encryption password (several commercial softwares, such as Passware Forensics Breaker and Elcomsoft Phone, support this feature). Obviously, the success of the attack depends on the usual variables such as the complexity of the password, the number of attempts in a time unit (e.g. seconds), the availability of an additional (custom) wordlist (e.g. other passwords). If the user of the device has an Apple computer, it can be possible to try to extract the password from the backup of the keychain, also in this case, knowing the user's password, or by making an attack.

If the attempted attack on the password of the backup fails, it is in any case possible to view the list of files inside the backup with their metadata, analysing the file manifest.mbdb that, as already explained above, is cleartext even when the backup is encrypted. In this regard you can use the script mbdbls developed by Hal Pomeranz [4].

/ THE ANALYSIS

As underlined in the previous section, the analysis of data acquired through backup is obviously strongly influenced by the possibility to access the contents of the extracted backup.

When the backup is in clear or encrypted but the password is known, you can proceed with the analysis of important information inside it. The iOS operating system basically uses two types of files for storing configuration and data of each application; plist files and SQLite databases.

We can group the files into three macro categories:

- System configurations file
- Native applications data and configuration files
- Third-party applications data and configuration files

Some of the most relevant configuration files are: →

/ MORE INFO

M. Epifani and P. Stirparo, Learning iOS Forensics, Pack Publishing 2015.
The most updated on the subject.

A. Belenko; "iOS Forensics with Open-Source Tools" at Zeronights 2014, http://2014.zeronights.org/assets/files/slides/belenko.pdf"
An overview on how to do iOS Forensics with open sources tool.

J. D'Orazio, iOS Forensic, 2013: https://wiki.cis.unisa.edu.au/wki/images/7/7c/DORAZIO_iOS_Forensics_Final_Revise.pdf
How to perform file carving of image files from an iOS system.

- **SystemPreferencesDomain/SystemConfiguration/com.apple.accounts.exists.plist:**
  Contains the accounts configured in the device and grouped by type (e.g. Apple, Google, Facebook, Email, etc.)
- **HomeDomain/Library/Accounts/Account3.sqlite:**
  Contains details of the accounts configured into the device (e.g. username and type of stored credentials like password, OAuth, etc.)
- **SystemPreferencesDomain/SystemConfiguration/com.apple.wifi.plist:**
  Contains information about the Wi-Fi networks configured in the device
- **WirelessDomain/Preferences/com.apple.commcenter.plist:**
  Lists the telco provides in use and other information about the SIM card
- **HomeDomain/Library/MobileBluetooth/com.apple.MobileBluetooth.ledevices.plist:**
  Contains information about the Bluetooth devices paired with the iOS device.

iOS is shipped already with several native applications pre-installed, which have data stored inside the backup as well.

Address Book

- **HomeDomain/Library/AddressBook/AddressBook.sqlitedb**
  Contains information about user personal contacts such as names, phone numbers, emails addresses, etc.
- **HomeDomain/Library/AddressBook/AddressBookImage.sqlitedb**
  Contains the images associated with the contact in the Address Book.

Calendar

- **HomeDomain/Library/Calendar/Calendar.sqlitedb**
  Contains all the user's calendars and associated events.

Call History

- **HomeDomain/Library/CallHistoryDB/CallHistory.storedata**
  Contains the list of outgoing, incoming and missed calls made through the cellular network as well as FaceTime.

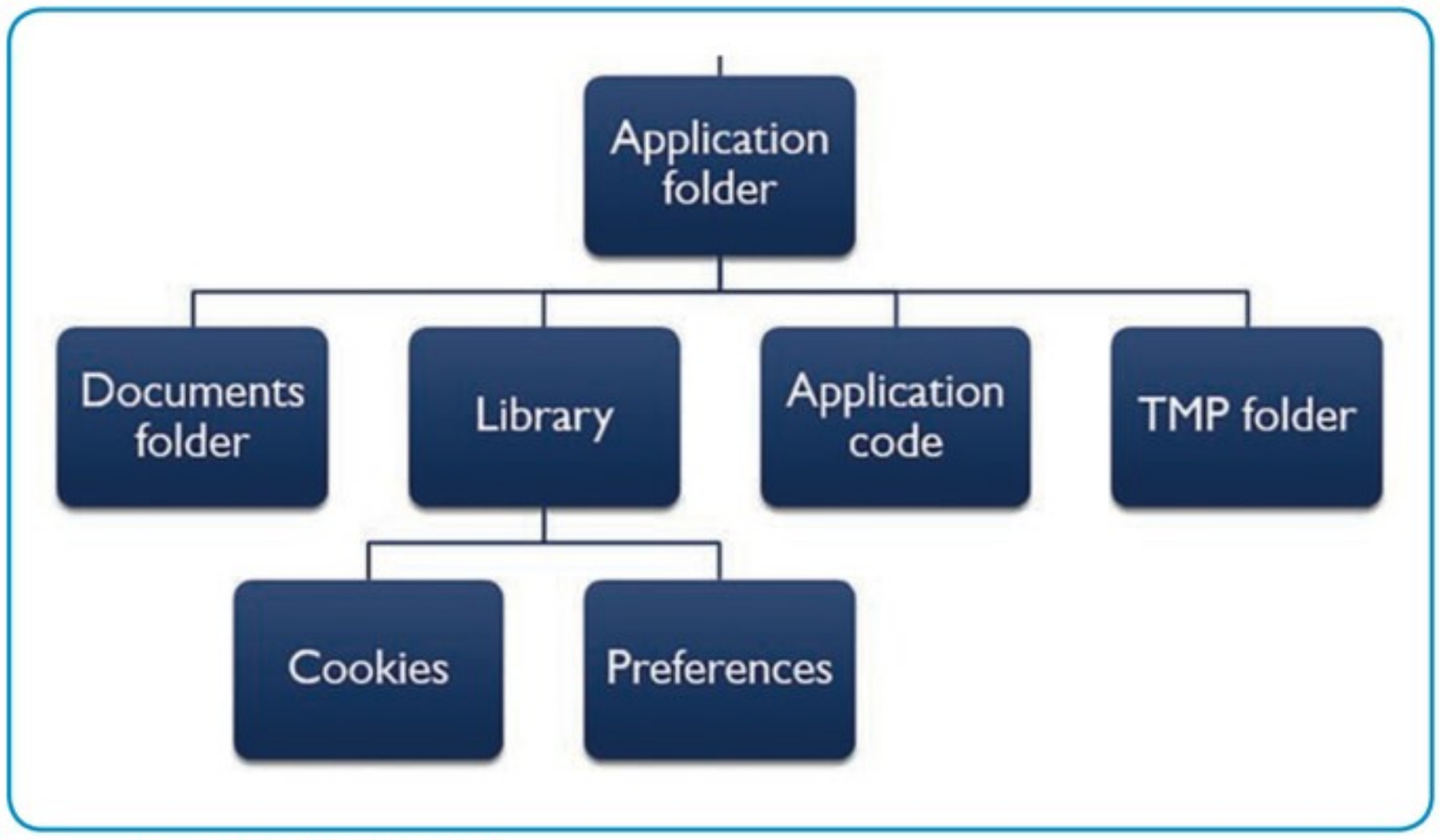


Figure 1. Specific Application Folders

SMS/MMS

- **HomeDomain/Library/SMS/sms.db**
  Contains SMS and iMessages sent and received, including the date it has been received, read and delivered (for iMessage), the text and the type (sent/received).
- **HomeDomain/Library/SMS/Drafts**
  Contains drafts of both SMS and iMessage.
- **MediaDomain/Library/SMS/Attachments**
  Contains the attachment received via MMS.

Apple Mail

- **HomeDomain/Library/DataAccess**
  Contains the email addresses of the account configured and their folder structure.
- **HomeDomain/Library/Mail/Recents**
  Contains information about the receivers and sent date of emails recently sent.

Notes

- **HomeDomain/Library/Notes**
  Contains the saved Notes, grouped by account.

Apple Safari

- **HomeDomain/Library/Safari/bookmarks.db**
  Contains Safari Bookmarks

/ APPLICATION ANALYSIS

Sometimes you may need to investigate an application suspected to be malicious. To begin with, you need to jailbreak your testing device in order to have full control of it, being able to access all kinds of information: stored, in the memory or transmitted over the network. Once the device is jailbroken and Cydia installed, you also need to install OpenSSH and BigBoss recommended tools, a package containing utilities such as apt, make, wget, etc.

To connect over ssh to your iPhone via USB, edit the ~/.ssh/config file in your computer by adding the following:

```
Host usb
HostName 127.0.0.1
Port 2222
User root
RemoteForward 8080 127.0.0.1:8080
```

The usb hostname is now properly mapped to the ssh connection, while the last row will set up port forwarding so that connections to iPhone's port 8080 will be forwarded to port 8080 locally on the computer, and you can intercept the network communications via proxy. Finally, you need usbmuxd to listen on port 2222. This daemon is in charge of multiplexing connections over USB to the iOS device. To complete the procedure on OSX, you can simply use the following command:

```
$ brew install usbmuxd
$ iproxy 2222 22
$ ssh usb
```

Now you have a shell in your iPhone via USB and can go ahead installing and utilizing all command line tools of your choice.

"AT THE END OF 2015 THERE WERE ABOUT 1.2 MILLION APPLICATIONS AVAILABLE."

- **AppDomain/com.apple.mobilesafari/Library/Safari/History.db**
  Contains Safari navigation history
- **AppDomain/com.apple.mobilesafari/Library/Safari/RecentSearches.plist**
  Contains the most recent searches made through Safari.
- **AppDomain/com.apple.mobilesafari/Library/Safari/SuspendedState.plist**
  Contains the status of all currently active tabs in Safari.
- **AppDomain/com.apple.mobilesafari/Library/Safari/Thumbnails**
  Contains the thumbnails in PNG format of the Safari currently active pages.

Photos

- **CameraRollDomain/Media/DCIM**
  Contains videos and pictures taken from the device camera or saved from other third-party applications (e.g. WhatsApp, Facebook, etc.). The pictures taken with the device camera are stored in JPG format while the videos are in MOV format. Moreover, it contains several subdirectories where the files are actually stored, which names contain an increasing number (i.e. 100APPLE, 101APPLE, 102APPLE, etc.), and each of these folders may contain up to 1000 files.
- **CameraRollDomain/Media/PhotoData/MISC/DCIM_APPLE.plist**
  Contains information about the active folder and the number of files.
- **CameraRollDomain/Media/PhotoData/Thumbnails**
  Contains 4 files in the proprietary format ITHMB, one for each possible thumbnail size. Software like ITHMB Converter (http://www.ithmbconverter.com) are able to extract and convert such thumbnails into JPG format.

The analysis of third-party applications is obviously a bit more complex: just think that at the end of 2015 there were about 1.2 million applications available on App Store, according to statistics published by Appshopper.com. Application data within a backup are stored within the AppDomain path and each application has its own specific folder. Although the main folder structure of the applications is basically standard, as shown in Figure 1, developers can obviously add their own custom folders. Therefore it is important to check carefully the entire internal structure of each application.

If you are interested in an updated list of the main iOS artifacts, both for native and third party applications please visit the Mac and iOS Forensic Artifacts [5], which is where we plan to update our findings for iOS9.

CONCLUSIONS

With every release of new devices, as well as major versions of iOS, Apple keeps enhancing their mobile systems from a security standpoint. On the one hand, considering it from the user perspective this is for sure a good and interesting approach. However on the other hand, for the forensic analyst it increases the level of difficulty in the analysis but even more in the acquisition phase. As we have seen in this article, the new security features and the patches of previous vulnerabilities (some of which were exploited for some acquisition methods) have reduced the options available for a forensic acquisition of iOS devices to the backup/logical method for most of the cases.

We have highlighted four different scenarios that an analyst may face during the acquisition and some important actions he/she should do and not do (like pushing the home button in the case of an unlocked but passcode protected device). On the analysis side, with the well known native iOS applications it should still be fairly easy while the big challenge resides with third-party apps.

This is mainly for the two reasons that we have already introduced during the article. The first reason is that there are too many apps, which implies that the analyst may need to spend a lot of time on analysing each and every one of those he/she does not know about. To give an example, there are a lot of applications that allow users to exchange chat messages among them (and I'm not referring to messaging applications, but others like games, etc.). If the analyst does not know that the application has such feature, he may overlook and miss the chat content inside the app. The second reason is that the structure is not fixed, but a developer may customize how the internal folder structure will look, making it difficult for the analyst to retrieve certain types of information.

REFERENCES

1. Apple; Apple iOS Security Guide, https://www.apple.com/business/docs/iOS_Security_Guide.pdf
2. J. Zdziarski; Identifying back doors, attack points, and surveillance mechanisms in iOS devices, http://www.zdziarski.com/blog/wp-content/uploads/2014/08/Zdziarski-iOS-DI-2014.pdf
3. libimobiledevice, http://www.libimobiledevice.org/
4. H. Pomeranz; mbdbls, https://github.com/halpomeranz/mbdbls
5. P. Stirparo; Mac4n6 Artifacs, https://github.com/pstirparo/mac4n6

AUTHOR BIOGRAPHY

Mattia Epifani is partner and founder at REALITY NET – System Solutions, where he works as a senior consultant in Digital Forensics, Forensic Readiness, Mobile Security and Incident Response. He obtained a University Degree in computer science in Genoa (Italy) and a post-graduate course in Computer Forensics and Digital Investigations in Milan. He works as a digital forensics analyst for judges, prosecutors, lawyers and private companies, both as Court Witness Expert and Digital Forensics Expert. He is a regular speaker on Digital Forensics matters in different Italian and European universities and events. He is a member of DFA, IISFA, ONIF and T&L Center. Co-author of the book "Learning iOS Forensics" edited by PacktPub in March 2015.

AUTHOR BIOGRAPHY

Pasquale Stirparo is Senior Security and Incident Response Engineer at a Fortune 500 company. Prior to this, he founded SefirTech, an Italian company focusing on mobile security, digital forensics, and incident response. Pasquale has also worked at the Joint Research Centre (JRC) of European Commission as digital forensics and mobile security researcher, and contributor to the standard ISO27037. Pasquale holds a Ph.D. in Computer Security from Royal Institute of Technology (KTH), Stockholm, a M.Sc. in Computer Engineering from Polytechnic of Torino, and he has GCFA, GREM, OPST, OWSE, and ECCE certifications and is a member of DFA, T&L Center, ONIF. He is co-author of the book "Learning iOS Forensics".

BOOK REVIEWS

LEARNING IOS FORENSICS

Reviewer Name
Steve Waterson

Authors
Mattia Epifani & Pasquale Stirparo

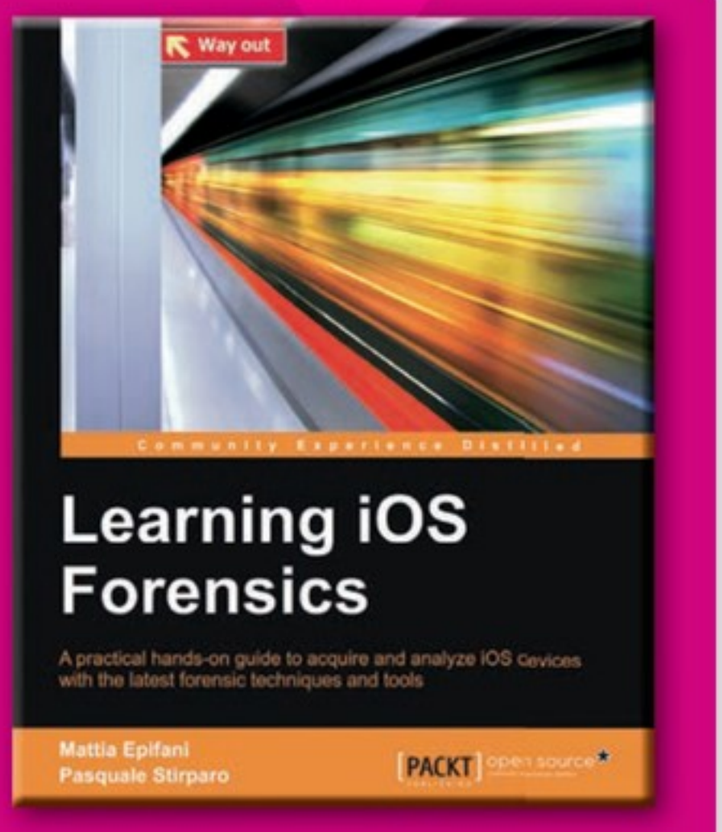
Publisher
[PACKT] Publishing

Date of Publishing
2015

ISBN
978-1-78355-351-8

Price
£24.99 / $39.99

Rating
★★★★★



The first thing that you notice with this book is that it is written by two Italian authors who have significant knowledge and experience that they have a desire to share, just their biographies alone make for interesting reading.

The book is divided up into 7 chapters and 3 appendices leading the reader in gently to the topic of mobile digital forensics and digital evidence in Chapter 1 to Applications and Malware Analysis in Chapter 7. At the end of each chapter is a summary followed by a set of self-test questions with the answers provided for checking in Appendix C.

Chapter 2 takes a look at the various iOS devices before looking at the file system, device and system partitions and other aspects of the operating system. This is followed by the introduction of tools and techniques for evidence acquisition from the iDevices using case studies to help explain the techniques used by example. The book is also littered throughout with pictures and screen shots that aid in the explanation of the tools and techniques.

Analysis of the native applications that are installed on all iOS devices and the third party applications that are most commonly loaded is carried out in Chapter 4 providing you with a map of where to find the most interesting artifacts. This is followed by an analysis of the iOS backup locations and finishes in Chapter 7 with malicious applications and malware analysis. All backed up with two appendices that are full of resource references and locations where tools can be found and downloaded.

/ CLOSING SUMMARY

This quick canter through the book does not do it justice; it is simple in construct and layout with simple yet effective explanations of the devices, the operating system and the applications. This book is a useful reference for both seasoned practitioners and beginners. Anyone with an interest in iOS Forensics will find this book useful.

"AT THE END OF EACH CHAPTER IS A SUMMARY FOLLOWED BY A SET OF SELF-TEST QUESTIONS WITH THE ANSWERS."