Kandidat rapport, Maj 2013

IT-Forensik och Informationssäkerhet

Forensic investigations of Apple's iPhone

Mats Engman

Kandidatuppsats

Sektionen för informationsvetenskap, data- och elektroteknik

Forensic investigations of Apple's iPhone

Kandidatuppsats

2013 Maj

Författare: Mats Engman

Handledare: Mattias Weckstén

Examinator: Urban Bilstrup

Sektionen för informationsvetenskap, data- och elektroteknik
Högskolan i Halmstad
Box 823, 301 18 HALMSTAD

Kandidatuppsats
Rapport, IDE
Sektionen för informationsvetenskap, data- och elektroteknik
Högskolan i Halmstad

Abstract

The use of smartphones has grown increasingly over the last few years. These devices contain much information that could be interesting during a police investigation.

One of the most used smartphones, to date, is the Apple iPhone. You can assume, if it's not already the case, that these devices will have a bigger significance when it comes to gathering evidence and information about a person's social connections and whereabouts. In this study I am going to perform three experiments based on different conditions we may face in forensic investigations, and gather certain information from the iPhone. I am also investigating what challenges this presents to us from a law enforcement point of view.

There are a couple of papers on this subject but most of them address older versions of iOS and iPhones only. I will be using iOS 6.0 and compare the different methods based on a couple of interesting data artifacts that could be potential evidence in criminal cases.

Table of contents

1 Introduction

According to market research presented in an article [1], the iPhone is one of the most common smartphones on the market today. As these devices grow in popularity, so does the interest in accessing all data these devices contain. The art of Mobile Forensics have over the last few years become an important part in the forensic community.

A smartphone is essentially a small computer, so many of the concepts of computer forensics can be applied here. There are however some important differences. Data on smartphones are extremely volatile. It is constantly changing (unless the phone is turned off). Additionally one cannot simply copy the contents of the memory, the data is encrypted and the operating system of the phone prevents us from running any applications that hasn't been signed by Apple. So we need to work our way around these obstacles.

With every release of a new phone or operating system, a new range of problems arise. So the battle of mobile forensics is a never-ending one.

1.1 Background

Statements from Law-enforcement is indicating that there is a need for more work to be done in this field, and with the increasing use of smartphones I deemed it important to write about this. With new advances, more and more commercial mobile phone forensic suites appear on the market, the problems of mobile forensics are being mitigated, but there is still work to do.

Smartphones today are primarily used to connect to people, through phone calls, social media, messaging and so on. So if we want to map a person's acquaintances and connections, the mobile phone is a goldmine for this purpose.

1.2 Problem statement

Today there are different ways of extracting data from an iPhone, depending on the conditions we are faced with during an investigation.

The purpose of this paper is to show how you can perform a forensic analysis of the memory in an iPhone and show the differences in the three most common methods used today.

The main questions to be answered are:

- How do you perform a forensic analysis of an iPhone?
- What differences are there in different extraction methods?
- What type of information do you want to examine?

1.3 Mobile forensics

Mobile forensics or mobile device forensics is a category of computer forensics that includes mobile phones, smartphones, PDA's and GPS's among others. These devices are built to be as small and portable as possible. These types of devices have certain characteristics. To keep the physical size of the memory small, a flash memory is used. There are different types of flash memory: NOR and NAND. Like mechanical hard drives, flash memory doesn't need power to maintain data on the chip. The iPhone contains a NAND chip. This memory is capable of storing more data than the NOR, but it is not as stable and it is cheaper. The NAND memory also needs a RAM memory to work. Flash memory has a more limited lifetime that other hard drive due to the "wearing" that erasing data does to the chip. There is a certain number of times a erasing can be performed before the chip breaks, or get wearied out.

Flash memory does present some problems for forensic investigations [2]. The memory has built-in garbage collection and fragmentation functions, this is to minimize the "wear" on the chip. These work different on different vendors and are sometimes poorly documented. The fact that the memory shifts data around and overwrite sectors or pages without the operating system controlling it makes them unpredictable. When acquiring a memory, we want to keep the data unchanged [3].

2 Ethic discussions

I will present different ways to extract data from an iPhone. Some of these methods involve finding ways to circumvent the security features of the smartphone. We face the same problems here as virtually any research in the security field, the experiments and could be used for malicious purposes as well.

This papers audience is IT-Forensic investigators, law enforcement officers and computer security technicians. No legal aspects will be covered as it's outside the scope of this paper.

3 Methodologies

I will use a qualitative approach and perform hands-on experiments in a lab environment. The results of the experiments will then be analyzed and compared to each other to lift the differences between them. As Zhang et al. are describing in their paper, a qualitative research approach is about conducting investigative experiments [4]. Traditionally this is done in the sociology and empirical studies but it is also applicable on information system experiments.

3.1 Review of similar studies

There are some papers published in the field of iPhone forensics, although most of them address older versions of iOS. Therefore there is a gap in the research on current versions of the phone.

Bader and Baggili are using a logical analysis method in their work on examining iPhone backups [5]. To perform a logical backup on the phone, a good method is to use the built-in function in iTunes. When we connect our iPhone to a computer or upgrade the firmware, the program will ask you to perform a backup.

Punja and Mislan's paper on Mobile Device Analysis uses a more descriptive methodology [6]. It's often the aim of a qualitative research to have a descriptive approach and to follow up with examinations to made observations. They identify what evidence we can expect to find in a mobile device.

The file system on an iPhone is similar to HFS+. Burghardt and Feldman have written a paper on using the journal in the file system to extract deleted files on the disk [7]. They are showing that when using a method of examining the journal file in the file system you can find copies of files that has been deleted and removed from the active catalogue file.

3.2 Data extraction

The first stage of this project is to list what kind of information we as forensic investigators are interested in extracting. What data is useful to us in an investigation? This is done to lay the groundwork for the extractions and what artifacts I will be looking for.

To answer how you perform an analysis of an iPhone, I will conduct 3 different acquisitions with different methods of an iPhone and explain how they are performed so that they can be repeated. In the first experiment I will analyze backup files of the iPhone. As Satish B. is showing, this method is proven to work on iOS versions as late as 5.0.1 [8].

In the second experiment I will be performing a live analysis by using a commercial available tool, commonly used by law enforcements. This is the XRY suite made by Micro Systemation, see Appendix A, [a]. I use this method and forensic application because they are common among law enforcement agencies.

The third and final method I will be using is the one of performing a live extraction by the use of open-source tools. The way to accomplish this is to actually use the jailbreak tools available online and then load a custom program that performs the actual data collection. This approach is similar to Zdziarski's scientifically proven method which he proves is successful on earlier versions of iOS [9] [10].

I will then compare these methods to outline the differences between them. The comparison will be done with files such as call logs, messages, GPS data, social media data, pictures and deleted files. Which of these artifacts I focus on will be specified in section four of this paper.

The lab-systems on which I will work on are the Kali Linux and Windows 7 operative systems. The device is an iPhone 4, 16GB with iOS 6.0.

4 Data of interest

As forensic investigators, we want to find information that can be used as evidence and to get to know the person behind the system. We want to know what other people this person knows and maybe locate known associates. As the smartphones primarily function actually is to connect you to your friends and people you are associated with, this is a potential goldmine to be examined.

Note that there is loads of more data that you can extract on the phone but the data I am going to focus on is some of the data that can be of interest for forensic investigators and data that previous papers have focused on. It would be a huge amount of data to address therefore I have chosen to focus on these artifacts:

Call logs (Library/CallHistory/call_history.db) – This is an obvious data source when examining a mobile phone. Here we can get a list of people that the suspect has been in contact with, as well as timestamp data.

Contacts (Library/AddressBook/AddressBook.sqlitedb) – Contacts mean both phone numbers and e-mail contacts. Today the e-mail contacts could be far more than the usual traditional phone contact. Many phones offer the function to merge these two lists with each other.

Messages (Library/SMS/sms.db) – Messages here include SMS, MMS and also instant messages which are pretty common nowadays by using a third party message-app on the phone. This enables the user to send messages for free over the 3g/4g network.

Media (Media/PhotoData/Photos.sqlite) – Smartphones are often used as cameras. The cameras on the phones are getting better and better and are pretty handy, thus the interest in extracting these pictures.

Internet History (Library/Safari/History.plist) – This information is useful to us because this lets us see what Internet patterns the person has. We want to see any recent Google searches and visited websites.

Facebook data – This is a very interesting source of data. Here we can find a whole lot of information about other people in our person's social circle. Facebook is maybe the most widely used social application at this time so this information is very useful to us.

By data I mean Facebook accounts, friends and maybe check-in locations.

Location data (Library/Caches/locationd/consolidated.db) – Location data can be very useful in mapping the person's movements. This includes Wi-Fi locations and stored map locations.

Deleted files – Deleted files is probably the hardest among the things I've listed here to extract. To do this we need a physical copy of the memory on the phone and then you need to carve files out of un-allocated space. But it's nonetheless a good source to try to find information from. Deleted pictures and/or messages could be very valuable evidence.

5 File System

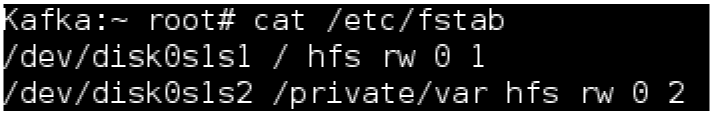
To better understand the forensic process of acquiring data from an iPhone, it's good to know a little about the file system that is used. iPhones and other iOS devices uses the HFSX file system which is a almost identical version of "Hierarchical file system Plus",(HFS+)[11].

The first few bytes of a HFSX volume contain the volume header of the HFSX file system. In this header, a couple of different fields exist. First comes the volume signature of the file system, this has the value of "`HX`". Then comes a version and an attributes field. After these two we have the lastMountedVersion field. In a HFSX volume this field has the value of "`HFSJ`". This means that this is a HFS journaling file system (every transaction to the drive is revorded). The header also contains fields like createDate, modifyDate, backupDate and fileCount among others [12] [13].

```
struct HFSPlusVolumeHeader {
    UInt16              signature;
    UInt16              version;
    UInt32              attributes;
    UInt32              lastMountedVersion;
    UInt32              journalInfoBlock;

    UInt32              createDate;
    UInt32              modifyDate;
    UInt32              backupDate;
    ...
```

The iPhone has two partitions as shown in Figure 1:



```
Kafka:~ root# cat /etc/fstab
/dev/disk0s1s1 / hfs rw 0 1
/dev/disk0s1s2 /private/var hfs rw 0 2
```

Figure 1. Shows the fstab on an iPhone after a jailbreak.

The `/dev/disk0` is the NAND chip of the phone. And here we have a root partition mounted under /, and a user partition under `/private/var`.

By default the root partition (`disk0s1s1`) is read-only, but after a jailbreak the iPhone's disk0s1s1 have the read-write permissions. This is essentially what jailbreaking is, modifying the fstab to mount the root partition as read-write.

The root partition contains the system files of the phone. This partition differs from 0.9GB up to 2.7GB. The rest of the phones memory is the user partition which contains all the user data.

There are mainly two ways that the data are stored in the phone, SQLite databases and binary lists called "property lists", (.plist)[14][15]. These lists have a XML type format and typically contain configurations and preferences.

Call history, messages, geo-locations and keychains are examples of data that is stored in the databases on the phone. To read this data we need a SQLite viewer.

6 Analysis

Now that we have established the data we are looking for it is time to do the actual analysis. This is the main section of the thesis. Here I will perform and describe the different analyses on the phone.

6.1 Backup analysis

Today there are different tools you can use to perform a backup of an iPhone. iTunes is probably the most common, but there are also forensic programs that can perform backups. When using iTunes, every time you upgrade the firmware on the phone, you are also required to take a backup of the phones state prior to the upgrade. You can specify in the settings in iTunes how often you will do backups.

The backups are stored in default locations depending of the operating system of the computer.

On Windows 7 the default path is:

```
%systempartition%\Users\%username%\AppData\Roaming\AppleComp
uter\Mobilesync\Backup\
```

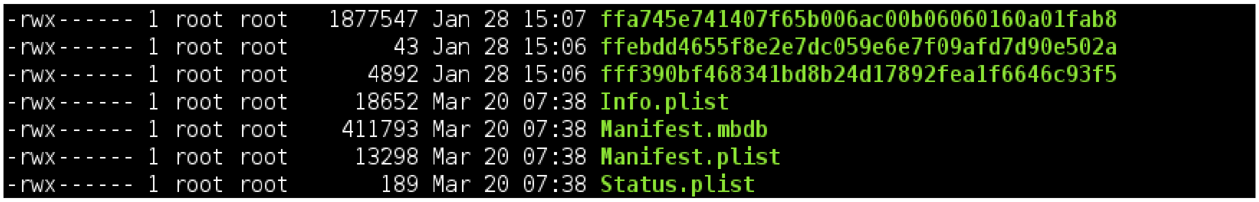
On Mac OS:

```
Users/%username%/Library/application
support/MobileSync/backup
```

The names of the files stored in the backup folder are a 40-digit long SHA1 hash value of the files domain and location in the file system. This makes that files unique identifier [16].

The name of the backup folder itself is also 40-digit long hash value. This is the phones UDID, Unique Device Identifier, and this is unique for every device.

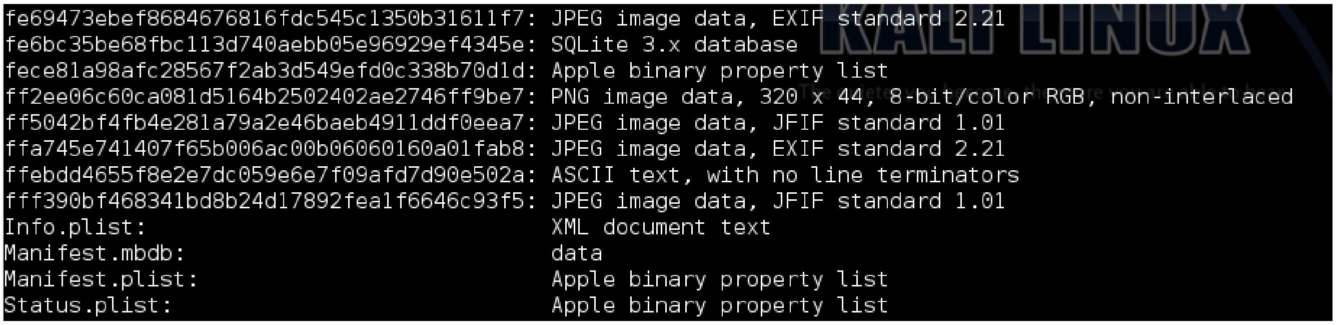
Once we browse through the backup directory we also notice that a lot of the files don't have a file extension (see fig 2).



```
-rwx------ 1 root root   1877547 Jan 28 15:07 ffa745e741407f65b006ac00b06060160a01fab8
-rwx------ 1 root root        43 Jan 28 15:06 ffebdd4655f8e2e7dc059e6e7f09afd7d90e502a
-rwx------ 1 root root      4892 Jan 28 15:06 fff390bf468341bd8b24d17892fea1f6646c93f5
-rwx------ 1 root root     18652 Mar 20 07:38 Info.plist
-rwx------ 1 root root    411793 Mar 20 07:38 Manifest.mbdb
-rwx------ 1 root root     13298 Mar 20 07:38 Manifest.plist
-rwx------ 1 root root       189 Mar 20 07:38 Status.plist
```

fig 2. File structure in backup folder.

We can determine the file types in Linux by issuing the command "file". See figure 3 for output of that command.



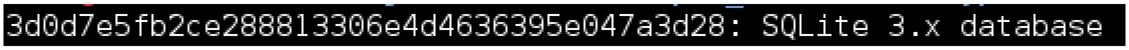
```
fe69473ebef8684676816fdc545c1350b31611f7: JPEG image data, EXIF standard 2.21
fe6bc35be68fbc113d740aebb05e96929ef4345e: SQLite 3.x database
fece81a98afc28567f2ab3d549efd0c338b70d1d: Apple binary property list
ff2ee06c60ca081d5164b2502402ae2746ff9be7: PNG image data, 320 x 44, 8-bit/color RGB, non-interlaced
ff5042bf4fb4e281a79a2e46baeb4911ddf0eea7: JPEG image data, JFIF standard 1.01
ffa745e741407f65b006ac00b06060160a01fab8: JPEG image data, EXIF standard 2.21
ffebdd4655f8e2e7dc059e6e7f09afd7d90e502a: ASCII text, with no line terminators
fff390bf468341bd8b24d17892fea1f6646c93f5: JPEG image data, JFIF standard 1.01
Info.plist:                               XML document text
Manifest.mbdb:                            data
Manifest.plist:                           Apple binary property list
Status.plist:                             Apple binary property list
```

Figure 3. Output of the Linux "file" command.

So, if we would like to manually examine, let's say the sms-messages on the phone, we would have to locate that database. The database is called `sms.db` and its home domain is `Library/SMS`. So we want to find the file with the file name with the SHA1-value matching "`Library/SMS/sms.db`", which in this case is:

3d0d7e5fb2ce288813306e4d4636395e047a3d28.

And by issuing the *file* command we see that this file is a SQLite 3.x database, as shown in figure 4.



```
3d0d7e5fb2ce288813306e4d4636395e047a3d28: SQLite 3.x database
```

Figure 4. File command output on the sms-database.

Now, what we need to do is to get a program that can read these databases and display the contents.

Examining the phones content in this manner would be a daunting task, luckily there are many applications available on the Internet that does this for us and presents the data in a good readable way.

When performing these experiments I tested a couple of them. I'm going to list some of them, essentially they do the exact same thing, but the look and the programming language in which they are created may differ.

Iphoneanalyzer – Iphoneanalyzer is written in Java and thus platform independent. It is licensed under GNU Public License v.3 (GPLv3). See Appendix A, [b], for a link.

We can browse the phones content and perform searches, both text and regular expressions, of the files.

Some of the features of the program stopped working with the release of iOS 6 and the developers are currently working on a solution.

An example of the differences is the Facebook application. On earlier versions of the application and iOS versions, a database of the user's friends was kept on the phone in: `com.facebook.Facebook/Documents/friends.db.`

As Mutawa et al. is confirming in his paper on investigating social applications on phones [17], when using iOS version 4 this database is present and should have the hash value of "`6639cb6a02f32e0203851f254`".

This seems not to be the case anymore. No such database was found in iOS 6.0, at least no on that location and by this acquisition method.

We can however see which account was logged in last on the application by viewing the "com.facebook.Facebook.plist" file. We can also get the session-key and last notification, last check-in, and how many unread messages there are in the inbox, on that account from this file [18].

IBackupBot – This program is available on Windows and Mac. There is a free version and a version that you need to pay for. The free version is not as packed with features as the above mentioned program. One good feature however is that you can open and read the contents of .plist files without using an external application (Appendix A, [c]).

So to sum up the result of this method's capability of finding the artifacts I listed as reference, see table 1.

Types of data	Able to extract?
Call logs	√
Contacts	√
Messages	√
Media	√
Internet History	√
Location data	
Facebook data	
Deleted files	

Table 1. Summary of backup analysis.

6.2 Analysis using the XRY forensic suite

The XRY forensic suite from Micro Systemation is a suite designed for mobile forensics. You get a kit with cable adapters for most new smartphones, mobile phones, and an analysis program capable of analyze the contents of the phones.

The suite contains software for exploiting smartphones to get access to the memory. The process of acquiring an image from an iPhone is very straight forward. You connect your phone with a cable to a computer and then run the acquisition program.

You can choose to do a logical or physical dump. I chose to perform a physical copy of the phones memory. The difference between a logical and a physical dump is that you get a lot more information with a physical bit-by-bit copy. We get the slackspace and un-allocated parts of the memory.

With a logical copy we can only get files that are allocated and supported by the operating systems synchronization feature.

Before anything can be done with the phone you need to put it in DFU mode (Device Firmware Upgrade mode). To do this you start the phone with the home and power buttons pressed simultaneously, and after about 9 seconds you release the power button but still with the home button pressed.

After this is done the program runs the exploit and uploads a custom image and kernel to the phone. This is run from memory and won't affect the data on the disk. If a passcode is enabled, the program starts to brute force it, as we need it to be able to unlock the phone. This process takes about 15-20 minutes with a four digit passcode.

Now the actual acquisition starts. The memory is dumped, a decoder runs and file signature analysis is then performed on the physical dump.

Once the acquisition is done we get a file with the .xry format which is a Micro Systemation proprietary format. We can now analyze the dump in XRY Reader. See fig 5 for a screenshot of the program.

By parsing the consolidated.db the program can derive some locations the mobile has been, though very limited. Back in 2011 a couple of security researchers found out that the consolidated.db file, which was new in iOS 4 (released in June 2010), was constantly recording the position of the phone. The media attention this got was huge and this raised a privacy discussion. Later Apple released a fix that addressed this issue and the phone stopped recording the locations. For the official apple response see Appendix A, [d].

Wi-Fi locations are however still stored in this database.

Under the locations tab in XRY we also find locations that other applications have stored. For example Apple maps and Training apps like Runkeeper. When you run Runkeeper, it stores the position of the phone every 3 second. This leaves a lot of location entries to browse through.

If you search for a location in Apple Maps that search and the location of that place is also recorded and stored on the device. This could give investigators useful information of places the suspect is planning on visiting etc.

As Zandbergen mentions in his paper, the phone uses a couple of different systems for determining its location. Depending on the method, the accuracy is more or less precise. We can't rely on the positions as much as we could consumer grade GPS devices, and depending on the location (indoors, outdoors) the precision will be different [16]. However, we can still get a pretty precise location. If the coordinates is with five or more decimals doesn't really matter to us as long as we can establish a fairly accurate location.

The location of the phone isn't just stored for WIFI or 3rd part applications. It could also be stored in the exif-data in pictures and videos taken from the device. Note that the phone only stores this information if the setting "location services" are turned on for the built-in camera application. There are ways for a malicious person to edit this information in order to slip detection or destroy evidence as Lallie et al. demonstrates in their paper on challenging the reliability of iPhone geo-tags [19]. This is an important issue that we must keep in mind during a forensic investigation.

By using this acquisition method we are able to derive more Facebook data. The Friends.db, as I mentioned in the previous chapter, could be found here by performing a search. In this case it was located in "Data/mobile/Applications/ED54E6D5-E2D3-47F0-8A35-2EFE0C22EE8C/Library/Caches/4110.0".

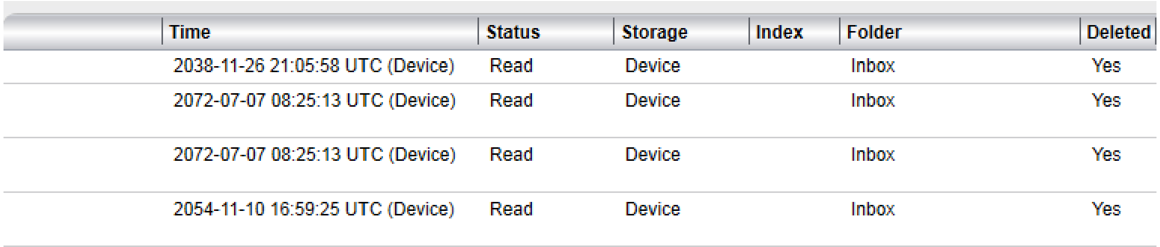
Here all the friends of the current Facebook account are listed.

See table 2 for a summary of this methods extraction.

Types of data	Able to extract?
Call logs	√
Contacts	√
Messages	√
Media	√
Internet History	√
Location data	√
Facebook data	√
Deleted files	Some*

Table 2. XRY analysis

*We can to some extent see deleted files (as it was a physical copy). In the SMS tab for instance, there is a column called "deleted", and that is set if the file was found to be deleted, see figure 5. Note here though that the information such as time and sender really can't be reliable as we see in the picture. That particular information could have been overwritten by some other data and thus display virtually anything.



	Time	Status	Storage	Index	Folder	Deleted
	2038-11-26 21:05:58 UTC (Device)	Read	Device		Inbox	Yes
	2072-07-07 08:25:13 UTC (Device)	Read	Device		Inbox	Yes
	2072-07-07 08:25:13 UTC (Device)	Read	Device		Inbox	Yes
	2054-11-10 16:59:25 UTC (Device)	Read	Device		Inbox	Yes

Figure 5. Deleted Files, shows files marked as deleted in the XRY suite.

This deleted tab can also be found at Notes, calendar, voicemail entries. We can't however browse the unallocated space and manually try to carve other types of files.

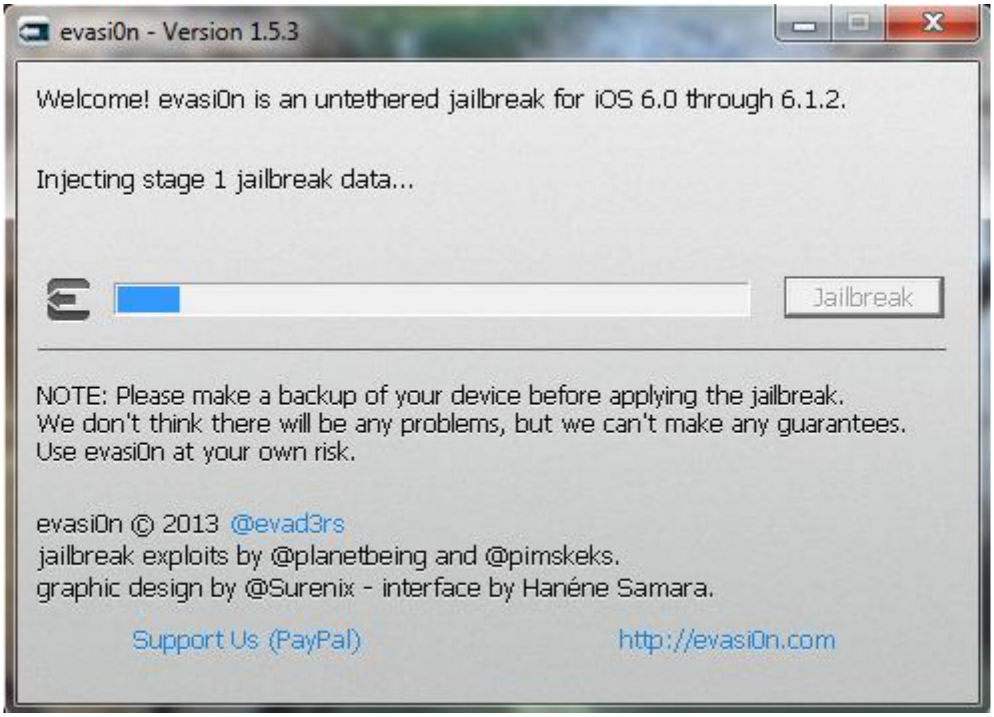
6.3 Analysis using jailbreak

To gain access to the phones memory, one approach is to jailbreak the phone. As described in the File System chapter, when we perform a "jailbreak" we use a security flaw on the phone to be able to exploit it and gain read/write permissions to the root partition.

The first step of jailbreaking is to find a tool that can exploit the current version of iOS on the phone.

When doing this we heavily contaminate the phone. This is important to understand when going with this approach. If we know what changes the procedure is introducing to the phone, we can always defend it, and explain as to why we chose to use this technique. Databases and files such as call history and messages and so on shouldn't be affected. To prove the level of contamination is an interesting topic for future papers, and we must be aware that contamination is in fact happening.

This experiment was done with the Evasi0n jailbreak application. To perform this, connect the phone to a computer and run the Evasi0n software. You will be prompted with a window as in figure 6.



evasi0n - Version 1.5.3

Welcome! evasi0n is an untethered jailbreak for iOS 6.0 through 6.1.2.

Injecting stage 1 jailbreak data...

NOTE: Please make a backup of your device before applying the jailbreak.
We don't think there will be any problems, but we can't make any guarantees.
Use evasi0n at your own risk.

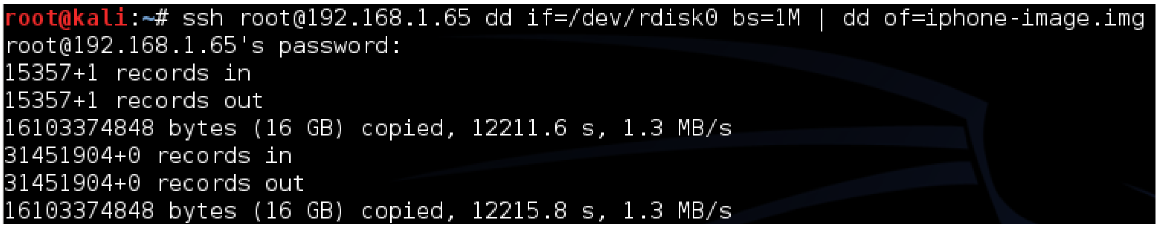
evasi0n © 2013 @evad3rs
jailbreak exploits by @planetbeing and @pimskeks.
graphic design by @Surenix - interface by Hanéne Samara.

Figure 6. Jailbreaking the phone, screenshot taken during the procedure.

When the jailbreak is done, Cydia will be downloaded and installed in the phone. Cydia is an application for searching and installing other applications that aren't signed by Apple and available on Appstore.

Through Cydia, download and install an SSH-client on the phone. This will be used to connect to it from a computer. When connected using an SSH-tunnel through a WI-FI network, we can browse the file system, look at any file without any restrictions.

To create a copy the disk, one simple command is all we need, using the dd-tool, see fig 7 for output from the tool.

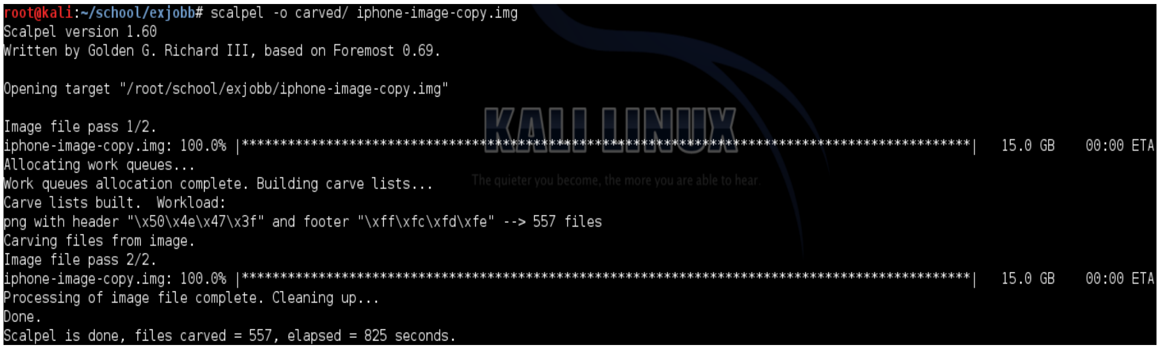


```
root@kali:~# ssh root@192.168.1.65 dd if=/dev/rdisk0 bs=1M | dd of=iphone-image.img
root@192.168.1.65's password:
15357+1 records in
15357+1 records out
16103374848 bytes (16 GB) copied, 12211.6 s, 1.3 MB/s
31451904+0 records in
31451904+0 records out
16103374848 bytes (16 GB) copied, 12215.8 s, 1.3 MB/s
```

Fig 7. Creating a disc-image using dd and ssh.

At this point we have an image file of the memory, also sometimes referred as hex-dump. Here we can do searches in the raw data of the image.

To carve files from a hex-dump or from unallocated memory we need to know the file header, and sometimes the footer, of the specific file type we are after. Take image files for example. When you take a photograph with the iPhone camera application the image is saved in the Portable Network Graphics (.png) format. The header of these files has the hex value of: `89 50 4E 47 0D 0A 1A 0A`.And by searching for this header in the memory we find the start of these image files. We then make an assumption that the data following this header is the data of the picture. By using this method we can find deleted files in the phones memory. See figure 8 for the program scalpel, carving images from the image file.



```
root@kali:~/school/exjobb# scalpel -o carved/ iphone-image-copy.img
Scalpel version 1.60
Written by Golden G. Richard III, based on Foremost 0.69.

Opening target "/root/school/exjobb/iphone-image-copy.img"

Image file pass 1/2.
iphone-image-copy.img: 100.0% |***********************************************************************************************|   15.0 GB    00:00 ETA
Allocating work queues...
Work queues allocation complete. Building carve lists...
Carve lists built.  Workload:
png with header "\x50\x4e\x47\x3f" and footer "\xff\xfc\xfd\xfe" --> 557 files
Carving files from image.
Image file pass 2/2.
iphone-image-copy.img: 100.0% |***********************************************************************************************|   15.0 GB    00:00 ETA
Processing of image file complete. Cleaning up...
Done.
Scalpel is done, files carved = 557, elapsed = 825 seconds.
```

Figure 8. Carving image files. Using the Scalpel tool to get pictures from the disc-image.

If we were to examine this image in a forensic application such as EnCase, DFF or FTK, we would get a file with no existing partitions and thus not be able to browse freely among the directories. To be able to do this we must try to recreate the partitions of the phone. Remember from chapter 5 that the iPhone contains HFSX partitions/volumes. To be able to recreate the volumes, we must find the sector, or "page" as is the correct term when talking about NAND memory, on the disk that contains the beginning of these volumes. And also as discussed in chapter 5 these volumes begin with a volume header. By performing a keyword search for "HX" (the signature in the volume header) and "HFSJ" (the lastMountedVersion), both without the quotes, the hits that contain both keywords in one and the same sector on the disk could assumingly be a HFSX volume header and thus the beginning of a partition.

One important thing here is that the volume signature is located two sectors in into the partition. So when recreating the partition we need to take the sector of the search hit minus two sectors.

I used EnCase when performing this, but this is not necessary. You can use your favorite forensic application.

The first hit to contain the keywords in the same sector gave me sector 32770 of the disk. By going to the disk view in EnCase and selecting sector 32768 (which is the starting sector of that volume). Then using the create partition option available and choose HFSX as file system. Click OK and EnCase should have created the partition. We can now browse the folder structure in this partition. As shown in table 3, we can locate all these files on the phone using this method.

Types of data	Able to extract?
Call logs	√
Contacts	√
Messages	√
Media	√
Internet History	√
Location data	√
Facebook data	√
Deleted files	√

Table 3. Analysis using jailbreak.

7 Discussion

We know how rarely users take backups of their data. This is no exception. If we find a backup of an iPhone, chances are that it is not that current. This is maybe the biggest drawback of backup analysis. And as the demand for data protection increases and encryption becomes more widely used, so will the encryption of backups. This is clearly a problem for these kinds of investigations. It is still however the default setting in iTunes to not encrypt the backup. This could easily be changed though, as it's a very simple configuration. The fact that we don't need access to the phone could be a huge benefit. When we investigate a person's computer, we should always look for these kinds of backups, even if the case isn't related with mobile forensics. This could be a great source of information about a person's associates. The procedure in extracting the data from a backup is relative simple and the amount of information that can be extracted is good. The use of cloud services such as iCloud will also have an impact, fewer will save their backups locally on a computer.

The analysis program that ships with the XRY suite is very easy to use and has a clear layout. As these kinds of forensic programs a fairly expensive we can demand more from them. The drawback here is that, to be able to perform the extraction, there must be a way to exploit the phones operating system, to get access to the memory. The company behind the program is developing these, but there is always a little gap between the release of a new iOS version and the release of an exploit for it. During this time we may not be able to perform any extraction.

The third method was by far the most cumbersome, but at the same time this opens up a wide range of opportunities that we could take advantage of. To Jailbreak the phone, I downloaded a jailbreak application, capable of jailbreaking the current iOS version, which were 6.0. I connected the phone to a computer and ran the software. When the process was finished I downloaded a SSH-client. To take an image of the memory I connected to the phone via SSH and used the Linux "dd" command. This gave me a bit-by-bit copy of the NAND memory. After this we can perform a wide variety of data extractions from the image. For example, I carved out image files by using the application Scalpel, importing it to a forensic software, such as EnCase or FTK.

Time is always a critical factor in investigations and the steps performed with the jailbreak method takes time. They are more advanced and time consuming. The level of experience of the investigator could impact the result here, as one with more knowledge probably could derive more data from an image like this. It's important to keep the knowledge fresh and educate the staff on this subject.

As in any forensic investigation we have to have an open mind about the data that we extract. There are ways to alter data and use evasion techniques, so called anti-forensics. Thomas Marryat et al. show an example of this in their paper on falsifying sms messages on mobile phones [20]. Altered timestamps on messages or call logs could be hard to detect.

In comparison to each other, the method in which I was using a well established mobile forensic suite, is less time consuming and it takes less work to produce some actual evidence from the phone. This method also doesn't contaminate the phone as much as manual extraction. But it is way more expensive. It is important to weigh these pros and cons against each other when determining the way you are going to examine a mobile phone.

8 Conclusion

The goal of this thesis was to provide an overview of methods to perform a forensic examination of an iPhone and to compare these different methods to each other, in terms of usability and ability to acquire certain files.

Files that are interesting to Forensic investigators and that could serve as potential evidence are Contacts, Messages, Pictures, Videos, Internet History, Location data, Social media data and deleted files.

I have in this paper shown that by performing an analysis of an iPhone's backup file, not all these files, or forensic artifacts, could be acquired. Location data, Facebook data and deleted files could not be found. However, as shown by using an application that reads the data in the backup files and present it to us, call logs, messages, contacts, pictures, videos and Internet history were found.

When using a commercially available forensic suite, XRY, I was able to extract more information than with the first method. By taking a physical copy of the memory I was also able to find some deleted files, such as SMS-messages, calendar entries and voicemail messages.

By Jailbreaking the phone and using available software online I was able to get a bit-by-bit copy its memory. And after this I used different tools to extract all the data. We can also find deleted files by, for example, carving the memory after known file-headers. I was able to find all the files using this method.

My work also shows that by manually jailbreaking the phone, and extract the data with different open-source tools, the time needed to perform these steps is higher than the two others. The easiest and least time-consuming method is by far the one using the XRY suite, followed by backup examination.

Table 4 shows a summary of the files that could be extracted from the different methods.

Types of data	Backup	XRY	Jailbreak
Call logs	√	√	√
Contacts	√	√	√
Messages	√	√	√
Media	√	√	√
Internet History	√	√	√
Location data		√	√
Facebook data		√	√
Deleted files		Some	√

Table 4. Summary of extraction.

References

[1] ABI Research, “45 Million Windows Phone and 20 Million BlackBerry 10 Smartphones in Active Use at year end.” [Online], Available: http://www.abiresearch.com/press/45-million-windows-phone-and-20-million-blackberry

[2] Fiorillo S., “Theory and practice of flash memory mobile forensics”, in *Proceedings of the 7th Australian Digital Forensics Conference*, 2009, pp. 52-84, http://www.theosecurity.com/pdf/Fiorillo.pdf

[3] Breeuwsma M. et al., “Forensic Data Recovery from Flash Memory”, in *Small Scale Digital Device Forensics Journal, Vol. 1, No. 1, June 2007*, http://www.ssddfj.org/papers/SSDDFJ_V1_1_Breeuwsma_et_al.pdf

[4] Zhang, Y. and Wildemuth, B. M., “Qualitative analysis of content”, in *Applications of Social Research Methods to Questions in Information and Library Science*, 2009, pp.308-319. Westport, CT: Libraries Unlimited, https://www.ischool.utexas.edu/~yanz/Content_analysis.pdf

[5] Bader M. and Ibrahim Baggili, “iPhone 3GS Forensics: Logical analysis using Apple iTunes Backup Utility”, in *Small Scale Digital Device Forensics Journal,* VOL. 4, NO.1, September 2010, http://securitylearn.net/wp-content/uploads/iOS%20Resources/iPhone%203GS%20Forensics%20Logical%20analysis%20using%20Apple%20iTunes%20Backup%20Utility.pdf

[6] Punja S., “Mobile Device Analysis”, in in *Small Scale Digital Device Forensics Journal,* NO. 1, June 2008, http://www.ssddfj.org/papers/SSDDFJ_V2_1_Punja_Mislan.pdf

[7] Burghardt A., “Using the HFS+ journal for Deleted file Recovery”, in *Digital Investigation 5*, 2008 P. 76–82, http://www.dfrws.org/2008/proceedings/p76-burghardt.pdf

[8] Satish B., “Forensic analysis of iPhone backups”, *Securitylearn.net,* http://www.securitylearn.net/wp-content/uploads/papers/Forensic%20analysis%20of%20iPhone%20backups.pdf

[9] Zdziarski J., *iPhone Forensics: Recovering Evidence, Personal Data, and Corporate Assets*, 2008.

[10] Satish B., “iPhone Forensics – On iOS 5 devices”, *Securitylearn.net*, http://www.securitylearn.net/wp-content/uploads/papers/iphone%20forensics%20%28iOS%205%29.pdf

[11] LeGault L., “HFS+: The Mac OSX file system”, February 22, 2009, http://pages.cs.wisc.edu/~legault/miniproj-736.pdf

[12] "HFS Plus Volume Format", *Apple Technical Note TN1150*, 2004, https://developer.apple.com/legacy/mac/library/#technotes/tn/tn1150.html

[13] Craiger P. and Burke P., "Mac Forensics: Mac OSX and the HFS+ file system", National Center for Forensics & Department for Engineering Technology, University of Central Florida, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.6018&rep=rep1&type=pdf

[14] Mac Developer Library, "Plist Manual Page", 2003, https://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man5/plist.5.html

[15] Carpene C.," Looking to iPhone backup files for evidence extraction", *The 9th Australian Digital Forensics Conference*, December 2011, http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1091&context=adf

[16] Mutawa N. et al,"Forensic analysis of social networking applications on mobile devices", in *Digital Investigation 9 journal*, 2012, P.24-33, http://www.dfrws.org/2012/proceedings/DFRWS2012-3.pdf

[17] Wong K. et al, "Facebook Forensics", Valkyrie-X Security Research Group, July 2011, http://boanchanggo.tistory.com/attachment/cfile7.uf@122225474E1D7750066451.pdf

[18] Zandbergen P., "Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning", in *Transactions in GIS*, 2009, 13(s1), P. 5–26, http://www.paulzandbergen.com/PUBLICATIONS_files/Zandbergen_TGIS_2009.pdf

[19] Lallie H., "Challenging the Reliability of iPhone Geo-tags", in The International Journal of Forensic Computer Science, 2011, P. 59-67, http://www.ijofcs.org/V06N1-FULL.pdf#page=61

[20] Marryat T., "Falsifying SMS Messages", in *Small Scale Digital Device Forensics Journal,* Vol. 4, No.1, September 2010, http://www.ssddfj.org/papers/SSDDFJ_V4_1_Marryat_Corcoran.pdf

Appendix A - Internet links

[a] Micro Systemation XRY [online], Available: http://www.msab.com/

[b] iPhoneanalyzer, [online], Available: http://sourceforge.net/projects/iphoneanalyzer/

[c] iBackupBot [online], Available: http://www.icopybot.com/itunes-backup-manager.htm

[d] Apple Press Info, “Apple Q&A on Location Data”, [online], Available: https://www.apple.com/pr/library/2011/04/27Apple-Q-A-on-Location-Data.html