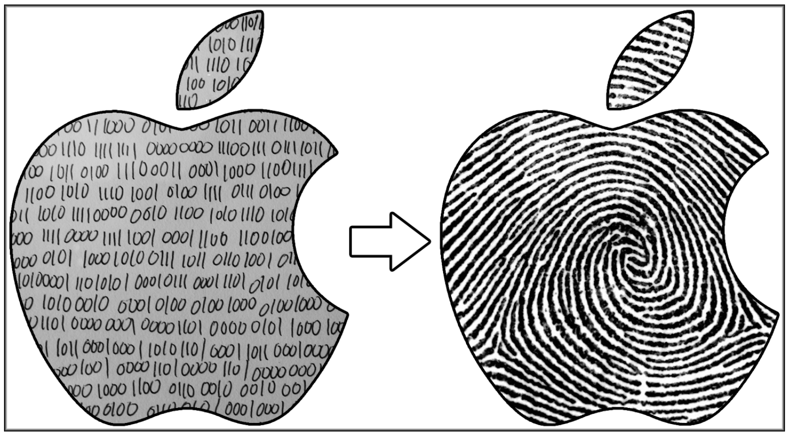
Universiteit van Amsterdam

System & Network Engineering

Handling iOS encryption in a forensic investigation

Research Project 2



Student

Jochem van Kerkwijk
jkerkwijk@os3.nl
#5631394

Supervisors

Coen Steenbeek
CSteenbeek@deloitte.nl

Marco Veen
MVeen@deloitte.nl

Derk Wieringa
DWieringa@deloitte.nl

July 19, 2011

Final version, rev. 2

Abstract

Since the introduction of the iPhone 3GS Apple offers a new form of encryption scheme called Data Protection. In this research project a relation has been made on how Data Protection has consequences for a forensic investigation. The report holds a literature study that elaborates on what has been done in this field before as well as to what the latest forensic methods are.

A practical attack as presented by Bédrune and Sigwald[4] has been performed on a first generation iPad equipped with version 4.3.3 of iOS, which currently is the latest version. Using this implementation it is relatively easy to circumvent iOS Data Protection using a bootrom exploit.

With these technical abilities forensic possibilities are identified. iOS devices are hardened against human attack vectors, not against a brute forcing computer. Data Protection has successfully been broken which allows for full forensic investigation.

The difficulty is in presenting this data to court, as the operating system on iDevices is prone to changes which introduce inconsistencies if the audit is performed by a different party. Traditional forensics simply does not apply to mobile devices and requires an alternative investigation path.

Acknowledgements

I would like to thank my supervisors Coen Steenbeek, Marco Veen and Derk Wieringa from Deloitte Risk Services for their time, effort and insight put into this research project. Additionally I would like to thank the colleagues at Risk Services for the warm stay, as well as the opportunity to use hardware and licensed software available at Deloitte.

– Jochem

Contents

1 Introduction **4**

1.1 Research question 5

1.2 Scope 5

1.3 Report structure 5

2 Apple iOS **6**

2.1 Operating System 6

2.2 Data Protection 6

2.2.1 Hardware Encryption 7

2.2.2 Software Encryption 7

2.3 Jailbreaking 7

2.3.1 Bootrom exploits 8

2.3.2 Legal issues 8

3 Computer Forensics **10**

3.1 Principles 10

3.2 Types of acquisition 11

3.3 iOS 12

4 Approach **15**

4.1 Workspace 15

4.2 Jailbreak 15

4.3 Custom environment 16

4.4 USB multiplexing 17
4.5 Bédrune and Sigwalds implementation 17
4.6 Defence Computer Forensics Laboratory Data Definition 17
4.7 Summary 18

5 Results **19**
5.1 Custom Environment 19
5.2 Circumventing Data Protection 19
5.3 Hex dump 20
5.3.1 cat acquisition 20
5.3.2 dcfldd acquisition 20
5.4 Image inconsistencies 20
5.5 Valuable information 21

6 Conclusion **22**
6.1 iOS and Data Protection 22
6.2 Circumvention of Data Protection 23
6.3 Forensic Aspects 24

7 Future work **25**

Chapter 1

Introduction

Since 2009[2] Apple is building their mobile devices with additional encryption hardware. Examples of these devices are the iPhone 3GS, iPhone 4 and both the first and second generation of the iPad[3]. Inside the operating system used on these devices, namely iOS which was previously known as iPhone OS, data is encrypted by default if a passcode is set. Apple calls this process *Data Protection*. Data Protection is the combination of using hardware based encryption in combination with a software keychain. This keychain is protected by a passcode and is also used to unlock the device every time the user wants to make use of the device.

When Apple presented Data Protection for the first time Zdziarski[28, 27] showed that it was relatively easy to gain access to confidential user data. With the introduction of iOS version 4 Apple attempted to tackle some of the pitfalls of their encryption scheme. However, recently it has become clear[8, 15] that there are some weaknesses regarding the keychain. This could be used for various purposes, as this is indirectly linked to the keys used to encrypt confidential data.

If this is combined with a "jailbreak"[25], a procedure to gain elevated rights, it should be possible to create a full disk image of the mobile device. However, forensic implications are unclear on this front. Elcomsoft[9] seems to have accomplished this feature, however due to the proprietary nature of the tool kit, the actual steps executed to perform the acquisition is unknown. This company might have built up a trust relationship with its customers but as this encryption scheme is new it does not automatically make it proven technology, certainly when it is impossible to peer review their methodology at a technical level.

1.1 Research question

Due to the above reasoning, the research question for this project is as following:

What are the forensic possibilities on an iOS device and what are the implications of "Data Protection" with respect to a forensic investigation?

To answer this question, the following sub questions are answered throughout this project:

1. What is Apple's "Data Protection"?
2. How does "Data Protection" work?
3. Can "Data Protection" be circumvented in order to access confidential data?
4. What are the possibilities to make forensic guarantees with regards to the acquisition and data-integrity of the evidence?
5. If it is possible to gain access to confidential data, what kind of information can be retrieved?

Additionally a reflection will be performed on the possibilities of anti-forensics for this procedure. Or in other words, what could a user have done to prevent a successful forensic investigation.

1.2 Scope

In this research the iOS device will be considered as the only source available for investigation and data extraction. There is a method known as a *escrow key attack* where one can use the backups made by iTunes as a source of key leakage allowing for easy access to the file system. This is a valid alternative route but will be disregarded.

1.3 Report structure

The report is structured as following. Chapter 2 and 3 will reflect on details regarding Apple's iOS and Data Protection, gaining access to iOS and forensic methods that must be adhered to. Chapter 4 will describe the approach and the practical steps executed. This includes the execution of the Bédrune and Sigwald methodology and the acquisition using `dcfldd`. Chapter 5 will present the results yielded from the mentioned approach. Finally, chapter 6 will give a conclusion and will also reflect on the outcome of the results and its forensic consequences.

Chapter 2

Apple iOS

In order to dive into the internals of the project, a literature study has been performed to better understand the underlying technologies which prepare the foundation of the practical approach performed in chapter 4.

This chapter begins with some details on Apple iOS, Data Protection, Jailbreaking and the attached legal issues.

2.1 Operating System

With the first generation of the iPhone, Apple introduced its mobile operating system *iPhone OS*. After the appearance of other devices such as the iPod and the iPad, the operating system is now more commonly abbreviated as *iOS*. iOS has evolved[26] from an OS that was just able to read mail, browse the web and was manageable through multi-touch gestures to an OS that fully provides for the digital (social) needs of current users. This varies from additional features like video conferencing and multitasking to user customization which allows for a different kind of device based per user.

Through this evolution and especially with topics like *bring-your-own-hardware* to work, more and more personal life as well as corporate information found its way into the device. This has led to an increased risk with respect to losing such a device and hence losing the user and corporate data available on it. This data will further collaboratively be called confidential data.

2.2 Data Protection

To protect the confidential data Apple introduced a form of encryption on their devices which they called *Data Protection*[3]. Data Protection follows the principles of RFC 3394[13][23] and can be separated in two parts, namely a physical part and a software part.

2.2.1 Hardware Encryption

Starting from the iPhone 3GS Apple is equipping their devices with encryption hardware. This hardware actually comes down to a dedicated AES coprocessor which limits the computational impact of encryption on the OS. In this coprocessor two keys are stored; a `UID` and a `GID`. The `UID` is often referred to as key `0x835`, which represents a unique AES key per device. The `GID` is a hardware key that represents a unique key per model.

2.2.2 Software Encryption

Each file or form of confidential data is encrypted with a unique key. This key is derived from the `UID`, `GID` and the public key of the process (or application) making the request and is stored in a so called *keybag*. This is a software container that on itself is protected by the *passcode* that is used to screenlock the device.

Usually this passcode is a four digit number. Besides the fact that this offers fairly little entropy (10^4 possibilities) and additionally often shows statistical distribution regarding probability[1] the passcode itself makes a stronger protection scheme than initially thought. Most importantly is the fact that data cannot be analysed (read brute forced) offline because decryption relies on the `UID` and `GID`.

Additionally, in the worst case scenario the user is able to initiate a remote kill switch. In this case the `UID` is destroyed. This procedure is effective as well as efficient, as with the removal of 256 bits all the encrypted files will be crippled.

2.3 Jailbreaking

Jailbreaking is the act in which the regular execution environment of an iOS device is granted elevated rights. This is considered useful as home-brew application will not run on iOS if they not got authorized to do so. Apple tries to prohibit this in order to gain control over the applications that are executed on iDevices and prevent against unwanted behaviour.

In order to execute a jailbreak, exploits are misused that are available in iOS. This can be through hardware drivers, the kernel itself and even side application such as the default browser installed.

There are three forms of jailbreaks available, namely:

Tethered Elevated rights are of temporarily basis and are lost after a reboot. Because of this, the device cannot boot on itself and needs to be jailbroken through a computer again to boot up properly.

Semi-tethered Similar to tethered with the exception that the system is able to boot up. However, elevated rights are still lost and require re-jailbreaking to regain.

Untethered Elevated rights are available even after a reboot of the device.

2.3.1 Bootrom exploits

A special form of exploit is the so called *bootrom exploit*. The bootrom is read-only-memory (ROM) and is the first stage of booting an iOS device. Apple secured this by signing each stage of the boot process. This is done once again to only allow known and trusted systems to be run on an iOS device.

The hacker community has succeeded in finding exploits in the bootrom, allowing for unsigned boot images. Apple is unable to fix these exploits through software because of the inherent properties of ROM. Through the perspective of the hacker community this is the most reliable and sustainable hack, as only hardware revisions from Apple can patch these exploits.

One of the most noteworthy bootrom exploits is *limera1n*. It is created by a hacker named George Hotz (aka geohot) and is currently still undisclosed. Geohot is also known for his legal clashes with Sony because of his successful attempts to gain elevated rights on the Playstation 3.

Besides geohot's own release, the exploit has also been incorporated into an open source project called *greenpois0n*. This project offers a clear and easy to use interface for using geohot's bootrom exploit, as well as an additional *injector module*. This module, named *Syringe*, is able to bootstrap into a jailbroken recovery mode or ramdisk, allowing the booting of unsigned images.

2.3.2 Legal issues

According to Apple jailbreaking was a violation of their copyright[16] as the process relied on a modified bootloader that was part of the copyrighted OS.

The Electronic Frontier Foundation (EFF) filed a request of exemption to the Copyright Office on grounds that "the culture of tinkering is an important part of our innovation economy". Users should be able to run the applications of their own choosing, and not let Apple decide such matter for the people. The exemption was granted, making the act of jailbreaking a legal procedure in the

US. Similar rights exists in the Netherlands through the "*Auteurswet*"[10]. This obviously does not implicate that any other illicit actions performed with the elevated rights are also legal.

Chapter 3

Computer Forensics

In order to answer the research question, another topic needs to be highlighted. As the main goal of this project is to perform the activities of gaining access to the encrypted confidential data in a forensic fashion topics such as computer forensics, forensic principles and also the relationship between traditional computer forensics and mobile devices will be elaborated upon.

As the details on the targeted set of devices is clear now, it is possible to put them into contrast to computer forensics. Forensics is defined as the application of a broad spectrum of sciences to answer questions that are of interest to the legal system[24].

3.1 Principles

The answers given to the legal system are through means of facts and probability and follow the principles seen in the scientific method. When ported to digital computer forensics similar steps are of interest. Even though computer evidence is usually used as a support factor it should still be handled appropriately. Main requirements are the rules of evidence, which can be specified as following[27][5][17]:

Admissible Evidence should be handled by an allowed official, as well as a technically apt official. This to allow the evidence in court and not to destroy evidence during acquisition as well as not to break the chain of custody.

Authentic The evidence needs to be properly connected to an entity or incident.

Complete The evidence should be delivered wholesome with respect to the incident in order to give an objective view of the events.

Reliable The method of acquisition should be built upon credible methodologies and scientific tests that can be reviewed by others.

Understandable and believable The evidence must be fully understood in order to present it. Also, the evidence should be clearly explainable to the (less technical) court.

These rules of evidence do not emphasise on any technical requirements. This is not unexpected, as each scenario has its own approach. However, more[22] can be said with respect to the imaging tool used for acquisition.

- The tool must create a forensic duplicate or mirror image of the original storage medium.
- Errors should be handled consistently and gracefully. If errors do occur, this should properly be logged and documented.
- The tool must not make any changes to the source medium.
- The tool must have the ability to be held up to scientific and peer-review.

The last bullet proves to be difficult, especially in the world of commerce. Often this is replaced by a “trust”- or “proven”-factor in order to protect the technical details of the process.

3.2 Types of acquisition

Besides the general rules that evidence acquisition should adhere to, there are also different technical types of acquisition. As not every digital device offers the same interfaces to communicate and interact with, various methods can be invoked in order to get data of these devices. This can be distinguished over 5 levels of acquisition methods[6].

Manual extraction Extraction of data is done by hand on the digital device itself.

Logical analysis The data is extracted through means of tools and interfaces provided by the device. Examples are Data cable connections, Bluetooth and Infra-red.

Hex dump Extraction is done at bit level. The result of a hex dump creates a identical copy of the data-carrier.

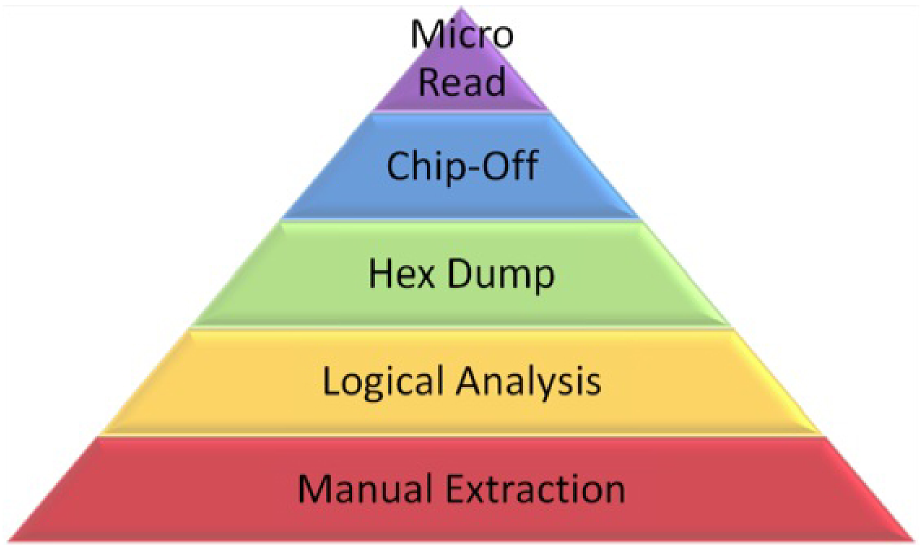


Figure 3.1: Forensic Acquisition Levels.[6]

Chip-off The extraction is performed by removing the data-carrier from the device and read out by placing it in a similar, but dummy environment. This is the case when hard drives are analysed offline or when a cellular devices wont boot due to broken hardware.

Micro-read Similar to chip-off, however the chip is not placed in a dummy environment but read through physically reading the gate status on a chip.

An overview of these types can be seen in figure 3.1. Going higher up in the image generally implies that more time, effort and costs are involved.

3.3 iOS

The main goal of this project is to perform computer forensics on a mobile device. There are however some main differences with relation to "traditional" computer forensics and mobile forensics. As smart phones are showing more and more properties similar to regular computers a lot of principles can be ported. However, new problems arise as well. One of the things mainly noticed[22] is that in general it is not wise to perform forensics on live systems. This is however sometimes required as chip-off or micro-read is not always an option because of its cost and effort.

The reason why this is considered as not wise is because of the fact that it is impossible to perform forensics on a running system without changing the state of the device. This is due to the uncontrollable nature of a running system. As most phones are also currently running propriety OS's a lot can happen under the hood. These changes are then only made visible through trial and error.

Another problem is additional artefacts created in order to perform the acquisition[12]. These changes are however sometimes required in order to gain access to the data. Implication is that these artefacts should be minimal, explainable and properly documented.

There are various applications and tool kits available[6] that perform forensics on iOS. However all of these tools are or not able to cope with Data Protection, or rely on the so called *escrow key attack*, which is not part of this research.

Nonetheless there are some successful methodologies, of which three will now be discussed. The first one is considered outdated (2008) but still proves to be useful, whereas the last two are fairly recent (2011). All of these methods apply the hex dump acquisition type.

Zdziarski methodology

Before the introduction of Data Protection Jonathan Zdziarski was already performing forensics on iOS devices. During this process he developed a methodology[27] to gain access to the file system and retrieve a hex dump from it.

At first, a custom environment is booted through a bootrom exploit. The environment itself only lives inside of volatile memory. This procedure is also called getting a ramdisk booted.

The environment is equipped with some custom tools. In order to communicate with the device, a SSH daemon is started. In addition, a client/server model is applied through a program called iRecovery. Through RPCs the forensic investigator is able to request a copy of the user partition, which is separated from the system partition.

Elcomsoft iPhone Forensics Toolkit

Recently Elcomsoft[8] succeeded in bypassing Apple's Data Protection allowing for forensic investigation. Elcomsoft released this product, named *iPhone Forensic Toolkit*[9], but is however not free and only available to forensic agencies and governments. Through their product page a video is made available showing the features offered by this tool kit. Two methods are possible. One is an attack through the backups created by iTunes. This attack is commonly known as the *escrow key* attack and offers the easiest way of gaining access to the encrypted file system as the UID is stored in the backup. Alternatively, Elcomsoft gained full access to confidential data through a side-channel attack. They circumvented the time needed to break the 256 bit AES key by means of just brute forcing the passcode on the iOS device itself.

Bédrune and Sigwald

In a similar time frame Bédrune and Sigwald presented their work at Hack in the Box 2011 in Amsterdam[4]. Their work is built upon a presentation given a year earlier at the same security conference[13]. In this project, which is open source and hence accessible to all, the two researchers cracked open Apple's Data Protection allowing for a fully decrypted file system and transparent access to the confidential data available on an iOS device.

Some similarities can be distinguished in the product video of Elcomsoft and the approach taken by Bédrune and Sigwald. They both make use of a form of the Zdziarski methodology and the custom ramdisk method. Also, in the video of Elcomsoft the use of popular jailbreak tool `greenpois0n` can be recognized[1]. `greenpois0n` is currently able to jailbreak any iOS device with the exception of the iPad 2. However, it will just be a matter of time before a bootrom exploit will be available for this device[2].

[1] *"Dear ElcomSoft, since you're using the GPLd greenpois0n code in your product, where can I download all your source code? #gplviolations"* – Zdziarski, Tweeted 7 Jun 2011

[2] As of July 6th 2011 an untethered jailbreak has already been released for the iPad 2[7]

Chapter 4

Approach

In this chapter the practical side of gaining access to confidential data is discussed.

The research question poses two main difficulties: first a way must be found to gain full decrypted access to the file system present on a iOS device. Second is using the available rights in such a way that it can be presented to a judge in court as evidence.

4.1 Workspace

To gain insight in the two problems a workspace was set up. As victim device a first generation iPad was used to experiment with. This is a tablet device that is able to run the latest version of iOS. The specifications of the device are described in table 4.1.

The hardware used on the attackers side was a regular laptop equipped with Windows 7. As iOS development is fairly difficult on non-Mac machines an virtual machine was set up with Mac OS X v10.6 and the iOS SDK for compiling application on the iDevice.

4.2 Jailbreak

In order to gain full access over the user space, elevated rights are required. To get these elevated privileges, a temporary jailbreak is executed on the device. This is done through the usage of *greenpois0n* (Section 2.3.1), which uses the underling *limera1n* bootrom exploit. This also allows for the booting of unsigned images.

Model	MB292ZP
Processor	Apple A4 (SoC)[1]; 1 Ghz ; ARM architecture
Memory	256 MB
Storage	16 GB
Radio	802.11a/b/g/n, Bluetooth 2.1+EDR, GPS, 2G & 3G
Latest iOS version	4.3.3

Table 4.1: Specifications of the victim: an iPad 1G

The full exploit delivers an untethered jailbreak. This is however an excessive result which is unnecessary as we are only interested in a temporary, tethered jailbreak. Luckily, the full exploit sequence can be aborted in order to achieve this.

4.3 Custom environment

With the ability to run elevated unsigned boot images, there is a possibility to run a so called custom ramdisk[27][20]. This is a temporarily host system that nests itself into the internal memory of the iOS device. This is especially useful with respect to forensic consistency, as the original OS is disabled, disabling unwanted changes that might taint the system.

There is a write-up on this topic[20] that explains the steps to create a minimal image based upon an original ramdisk. This original ramdisk is extracted from a Apple restore file which is used for factory resets of iDevices. It is bound to the version running on the iOS device. The ramdisk image is just a plain HFS+ file system which is native to Macs, making it fairly simple to add files to it. Supplementary, the write-up describes how to add some basic unix tools as well as a SSH daemon. This allows for communication with the device through a shell.

The implementation of the daemon is prone to the virus called *iWorm*, which abuses the default SSH root password `alpine`. As the connection is local this is not considered a problem with regards to possible tainting of the system. The reason behind this is that any forensic workspace should have been sanitized before starting an investigation.

[1]System on a Chip. Includes a CPU, GPU and volatile memory.

4.4 USB multiplexing

As networking capabilities are not enabled by default once booted into the ramdisk, a different way is chosen to communicate with the iOS device. By following the approach that Apple took with iTunes, USB multiplexing is used to communicate with the device. This is accomplished by using the `libusb`[11] library as well as python wrapper called `usbmuxd`[18]. Over this connection it is possible to establish a SSH connection and get a shell on the iDevice.

4.5 Bédrune and Sigwalds implementation

In order to circumvent the iOS Data Protection the implementation of Bédrune and Sigwald is used. The work is available through various tools and packages. The entire project is written in Python with the exception of some required binaries, of which the source code is also available.

The package includes shell scripts that can create a ramdisk, like earlier specified but dedicated to this specific project. A brute force application is also made available, in combination with a keychain dumper. The team has also made a variation available of the tool *HFS Explorer*, which is used to navigate and extract files from an HFS+ file system or image. The adaptation involves the read out of the keybag file that is extracted through the brute forcing application to be able to decrypt extracted files from the system.

In order to get a system image, a custom shell script is added to the ramdisk. This script invokes a tool called `netcat`, which is able to transfer data over TCP and UDP connections. The iOS device is set up as a `netcat` server and the client can request the file system from the iOS device using the same USB multiplexing technique as used with setting up the SSH connection.

4.6 Defence Computer Forensics Laboratory Data Definition

To change the default imaging process by Bédrune and Sigwald, which is just a `cat` of the user partition, but also to strengthen the imaging process with respect to forensics another tool is added to the ramdisk. This is the *Defence Computer Forensics Laboratory Data Definition*[14], which is more commonly known through its acronym `dcfldd`. This is an enhanced version of GNU `dd`, a tool to read and write to block devices in raw format. It adds additional functionality to `dd` which is helpful for forensic researchers. The main addition useful for this project is hashing-on-the-fly, split output as well as piped output.

Coming back to the requirements that an imaging tool should adhere to (section 3.1), `dcfldd` follows all of the guidelines.

- It creates a forensic duplicate image of the original storage medium by hashing its output instantly. In this approach MD5 algorithm and SHA1 algorithm was applied.
- In this experiment errors are handled consistently by writing zeroes in such a case.
- No changes are written to the source medium; (`dd` property).
- The tool can be held up for audit as it is completely open source.

4.7 Summary

To summarise the approach an overview can be seen in figure 4.1. A custom jailbroken state will be achieved by using the `greenpois0n` implementation. As soon as a custom environment is up and running, two steps are performed. The first one is brute forcing of the passcode, which will allow for access to the keybag file. Second is the hash dump of the user partition using `cat` or `dcfldd`. These will then be transferred through `netcat` in order to store the image on the investigators host machine. Finally, the file system will be decrypted using an adapted version of `HFS explorer` that makes use of the earlier retrieved keybag file.

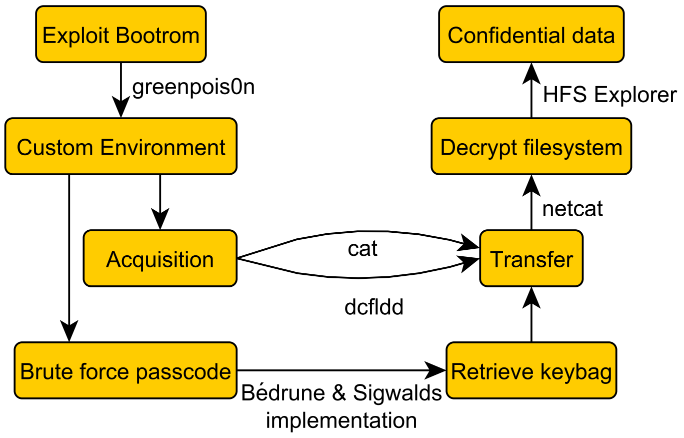


Figure 4.1: Overview of the approach.

Chapter 5

Results

In this chapter the results will be presented of the approach given in chapter 4. In short this includes booting a custom environment by using a bootrom exploit for quarantine, unwrapping decrypt functions by invoking a low level brute force attack on the iOS device, using `dcfldd` to make a forensically accepted hex dump of the user partition and using an adapted version of HFS Explorer to view the decrypted contents of the image.

5.1 Custom Environment

Using the available bootrom exploits as well as the custom ramdisk method passed successfully. A shell was accessible on the iDevice at root level.

5.2 Circumventing Data Protection

Using the implementation of Bédrune and Sigwald it was possible to brute force the passcode of the iPad. Initially their project was targeting iOS version 4.2.2 but can be confirmed to work successfully on version 4.3.3 as well. The time taken for this process came down to roughly 6 iterations per second, which translates to parsing the entire key space of 10000 elements in 30 minutes on an iPad 1G.

5.3 Hex dump

5.3.1 cat acquisition

The initial procedure of using `cat` in combination with `netcat` was able to dump the user partition. However, after a couple of dump attempts it was noticed that the images were not identical as they were showing hash differences that were created after dumping the images. Also, the slack space is not sent over so there is no possibility to carve for deleted files. Hence, this method of acquisition should be depreciated, as `dcfldd` does allow for the latter.

5.3.2 dcfldd acquisition

In order to get `dcfldd` functional on the iPad, recompilation was required. This was achieved by using the iOS SDK, which contains a dedicated iOS ARM compiler. Getting the so called *toolchain* together was a cumbersome and time consuming sequence of events. This due to the fact that compilation requires a compiler dedicated for iOS devices which is not freely available unless unsupported routes are taken. Nonetheless a functional binary was created.

`dcfldd` has a couple of output streams. The user partition is piped to the `netcat` like the initial procedure. However, again hashing differences were noted on two levels.

One was the difference between the hash that was available on the iPad created by `dcfldd` and a separate hashing tool on the host laptop. This would imply, that just like the *cat-method*, that the transfer method of using netcat leads to non-identical images. This was confirmed by the fact that the stream was not closed after the transfer was complete and was writing garbage. This can be fixed by using a different transfer method, such as SSH in combination with remote command execution.

The other level was hashing differences between several dumping attempts. This would imply that the system does change files. This is explainable through the methodology used to make the dumps. Between each attempt the system was booted normally to the point that the passcode was required.

5.4 Image inconsistencies

Using the available HFS explorer it was possible to compare the different dumps beyond hash level. By making use a directory compare tool the file differences were detected at content level. These changes vary from system changes, but also user land changes. An overview of these changes can be found in table 5.1.

Location	**Note**
System Files	
/.journal	FS Journal, changed
/db/dhcpclient/leases/en0-1,10_93_e9_56_6_d4	Wifi automatically connected
/log/notifyd.log	Additional daemon entry
/logs/keybagd.log	Additional daemon entry
/logs/lockdownd.log	Additional daemon entry
/preferences/SystemConfiguration/com.apple.AutoWake.plist	Log entry; binary
/preferences/SystemConfiguration/com.apple.PowerManagement.plist	Log entry ; binary
/preferences/SystemConfiguration/com.apple.wifi.plist	Log entry ; binary
/preferences/SystemConfiguration/preferences.plist	Computer link access time
/root/Library/Caches/locationd/cache.db	SQLite header change
/root/Library/Caches/locationd/cache.plist	Log entry ; binary
User Files	
/mobile/Library/AddressBook/AddressBook.sqlitedb	SQLite 3 Header change + 2 row changes
/mobile/Library/Logs/ADDataStore.sqlitedb	SQLite 3 2 line header change
/mobile/Library/Mail/Envelope Index	SQLite 3 header change
/mobile/Library/Preferences/com.apple.itunesstored.plist	Full change
/mobile/Library/Preferences/com.apple.MobileSMS.plist	SQLite 3 2line header change
/mobile/Library/Safari/Bookmarks.db	SQLite 3 Header change
/mobile/Library/SMS/sms.db	SQLite 3 header change
/mobile/Media/iTunes_Control/iTunes/IC-Info.sidf	Incremental binary log

Table 5.1: Overview of changed files.

5.5 Valuable information

As it was possible to decrypt the file system entirely, old forensic methodologies can be reapplied. As stated in the Zdziarski method[27], the list of data sources include:

- Keyboard caches containing usernames, passwords, and nearly everything typed on the iPhone.
- Screenshots of the last state of an application before the home button is pressed to return to the main menu.
- Deleted images, as slack space is available with `dcfldd`.
- Deleted calendar entries and contacts.
- A record of the last 100 calls made.
- Viewed Google Maps images and directions.
- Browser history and caches, even when deleted.
- Deleted email messages.
- Deleted voicemail.

As the iPad does not have calling functionality not all of these items were found. However, they are present in iOS even though unused. A good example of this is the tainted SMS database file after a reboot.

Chapter 6

Conclusion

What are the forensic possibilities on an iOS device and what are the implications of "Data Protection" with respect to a forensic investigation?

It can be concluded that forensic investigation is possible on iOS devices that do have Data Protection enabled. However, the acquisition phase of such an investigation has changed in contrast with previous[27] ones.

Mobile phones, or smart phones to that extend are getting more and more alike regular computers. Because of this, concepts with regard to computer security can be projected on mobile devices. Hence, Microsoft's security laws[19] are also getting more and more valid for mobile devices.

Law #3: If a bad guy has unrestricted physical access to your computer, it's not your computer any more.

6.1 iOS and Data Protection

The statement can be considered in the case for iOS devices as well. Throughout this research multiple sub questions were answered in order to conclude on the question if it is possible to do a "by court accepted" forensic investigation on iOS devices. By going through the concepts of Apple's Data Protection scheme and identifying weaknesses it seemed that Apple was missing the point of encryption, especially by seeing how easy it is to unwrap the decrypt function by merely brute forcing a four digit passcode. A note must be put in place that in this research no social investigation has been performed with respect to the (un)likelihood that users will not take the effort to switch to a more entropy rich unlock code. Either way, the recommendation is to enable the extended passcode. The length of this passcode should adhere to local security policies.

It however does become clear that these are not flaws by Apple, but architectural design choices between security and usability. Apple made the choice that data will be considered safe in its own iOS container/device; e.g. a chip-off attack will will not work, as the AES hardware register containing part of the decryption scheme is needed for decryption. Combined with a unlock pass code and the main feature of Apple products, which is simplicity and ease, they made a clear statement in their trade-off between security and usability.

Another point of interest is theft: Apple made it possible for users to remotely cripple their data if they lose their iOS device. This is achieved by executing a kill switch. If the device that is to be wiped is able to connect to internet, the switch is forwarded to the device which in turn erases the key present in the hardware register. This is an effective approach for anti-forensics as there is no need for computational and time intensive algorithms that permanently delete the confidential data.

In continuation of anti-forensics the user might have picked a custom protection scheme for encrypting confidential data as alternative to Apple's Data Protection. This will however require off-device key storage as the file system can fully be decrypted. But even in that scenario, the key for the custom container needs to be entered in the device in order to unlock it. As Apple also stores input of the keyboard, this can potentially also be retrieved from the appropriate cache file.

6.2 Circumvention of Data Protection

Because of the possibility to gain elevated rights at a bootrom stage it is possible to start up a custom environment. With this new environment the first level of Apple's defence is broken; the user data has not left the device and it was possible to interact with the hardware. Brute forcing the passcode is possible at this stage because the actual OS is not loaded. The main countermeasure by Apple, wiping confidential data after 10 invalid tries, is also circumvented by invoking function calls a level lower than the iOS API. Implementing this in the AES coprocessor will not resolve this issue because the keys are present on the file system, not inside the AES coprocessor. The `iphone-dataprotection` suite[4] is a free and open source initiative to technically allow investigators, or anybody else to that end, to gain access to the file system if only a numerical passcode is set.

These problems are probably also present in the iPad 2, as this device also runs iOS v4.3.3. However, due to the lack of a bootrom exploit the device is not affected by the attack.

6.3 Forensic Aspects

With the elevated rights combined with decryption possibilities customization is possible. Bérune and Sigwald[4] apply the methodology defined by Zdziarski[27], which in turn makes use of the custom ramdisk method described in [20].

To allow for a proper acquisition of the file system one has to cope with the difficulties of working on a live system. Luckily, a separation is made between the user data and system data through means of default partitioning. Even though imaging is prone to errors as getting into DFU mode requires a sequence of key combinations things are not as bad is it seems. In the worst case scenario some of the database files get touched and some header information gets changed. This does not alter the fact that there is an abundance of data available on iOS devices.

In order to use this data a proper documentation needs to be kept with regards to the acquisition process. The documentation, as well as the actual image should be stored in the case that a DFU boot attempt goes wrong and a manual inspection is required to see if an audit has been performed properly. In that stage the granularity has changed, as the hash of the partition changed and is not usable any more. Hashing each individual file will offer the solution here unless the file in question got changed during booting. In that case an textual/binary difference should be checked in order sustain forensic guarantees that the data or file did change, but for example only incremental.

During the execution of the approach, full file changes were discovered though. As long as these can be explained it should not pose a problem when discussed in court.

Making this believable and understandable in court will proof to be difficult. This is however the price of bleeding edge technology on which “regular” forensic methods simply do not apply. It will pay off, as the data that is being stored on iOS devices is extensive, including location data, conversations (email, voicemail, SMS) and searching history, and might just yield that piece of (support) evidence to make or break a case.

Chapter 7

Future work

As soon as the file system is imaged and decrypted, there are various files of interest available such as the mail database, the SMS database and also location history. All of these files are stored in a SQLite database format. In order to read these files a database viewer can be used, for example SQLite Database Browser[21].

However, these SQLite files are hard to work through as the user is presented with the raw layout of the database file (which is exactly the purpose of such tools). Unfortunately, this is not an efficient method for forensic investigators as these people are more interested in correlation between data. Main points of interest in that case are building time lines and being able to easily grasp sequences of events.

As the current tools are not meant for forensic investigators there is a need for improvement. To help the investigators more efficiently, a logical level needs to be built on top of the SQLite database files that does incorporate the needs of forensic investigators.

The architecture of such an application should also be decoupled from the data level, which is in this case specific to iOS. Following this approach a multifunctional forensic tool can be created which is not only dedicated to a single OS or mobile brand.

From a visual analytics point of view the application needs to able to easily explore the available (abundant) data. Hence, limitations of time-spans, specific contacts or specific forms of data (SMS, email, notes etc.) should form the main filter function. However, in this filtering process detail should not be lost and should incorporate elaboration functionality.

The visualisation can be extended with a dimension that identifies different types of confidential data through different nominal colour. With a click or hoover elaboration can be trigged to select and zoom in on the interested entity.

Bibliography

[1] Amitay, D. Most common iPhone passcodes, June 2011. [Online; accessed 14-June-2011]. Cited on page 7.

[2] Apple. iPhone [3GS] in Business: Security Overview, 2009. [Online; accessed 9-June-2011; Cached by wired.com]. Cited on page 4.

[3] Apple. iOS 4: Understanding data protection (HT4175), December 2010. [Online; accessed 9-June-2011]. Cited on pages 4 and 6.

[4] Bedrune, J. B., and Sigwald, J. iPhone data protection in depth, 2011. [Presentation; Hack in The Box Security Conference 2011 Amsterdam. Cited on pages 1, 14, 23, and 24.

[5] Brezinski, D., and Killalea, T. Guidelines for Evidence Collection and Archiving. RFC 3227 (Best Current Practice), Feb. 2002. Cited on page 10.

[6] Brothers, S. iPhone Tool Classification, 2007. [Online; accessed 9-June-2011]. Cited on pages 11, 12, and 13.

[7] Comix. JailbreakMe 3.0, July 2011. [Online; accessed 7-July-2011]. Cited on page 14.

[8] ElcomSoft Co. Ltd. ElcomSoft Investigates iPhone Hardware Encryption, Provides Enhanced Forensic Access to Protected User Data, May 2011. Cited on pages 4 and 13.

[9] ElcomSoft Co. Ltd. iPhone Forensic Toolkit, 2011. [Online; accessed 9-June-2011]. Cited on pages 4 and 13.

[10] Engelfriet, A. iPhone jailbreaken, mag dat nou toch, February 2009. [Online; accessed 3-July-2011]. Cited on page 9.

[11] Erdfelt, J. libusb, May 2010. [Online; accessed 7-July-2011]. Cited on page 17.

[12] Garner Jr., G. M. DD to Netcat NT Imaging, June 2002. [Online; accessed 14-June-2011]. Cited on page 13.

[13] Halbronn, C., and Sigwald, J. iPhone security model & vulnerabilities, 2010. [Presentation; Hack in The Box Security Conference 2001 Amsterdam. Cited on pages 7 and 14.

[14] Harbour, N. Defense computer forensics laboratory data defenition (dcfldd), February 2006. [Online; accessed 7-July-2011]. Cited on page 17.

[15] Heider, J., and Boll, M. Practical Consideration of iOS Device Encryption Security. Tech. rep., Fraunhofer, February 2011. Cited on page 4.

[16] Keizer, G. Apple: iPhone jailbreak hack violates the law, July 2011. [Online; accessed 3-July-2011]. Cited on page 8.

[17] Kubasiak, R. R., and Morrissey, S. *Mac OS X, iPod, and iPhone Forensic Analysis DVD Toolkit.* Syngress, 2008. Cited on page 10.

[18] Martin, H. usbmuxd, March 2011. [Online; accessed 7-July-2011]. Cited on page 17.

[19] Microsoft. 10 Immutable Laws of Security, 2011. [Online; accessed 9-June-2011]. Cited on page 22.

[20] msft_guy. Booting SSH ramdisk on new devices, June 2010. [Online; accessed 22-June-2011]. Cited on pages 16 and 24.

[21] Piacentini, M. SQLite database browser, June 2011. [Online; accessed 4-July-2011]. Cited on page 25.

[22] Prosise, C., Mandia, K., and Pepe, M. *Incident Response and Computer Forensics, Second Edition.* McGraw-Hill/Osborne, 2003. Cited on pages 11 and 12.

[23] Schaad, J., and Housley, R. Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394 (Informational), Sept. 2002. Cited on page 7.

[24] Wikipedia. Forensic science — wikipedia, the free encyclopedia, 2011. [Online; accessed 5-July-2011]. Cited on page 10.

[25] Wikipedia. IOS jailbreaking — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed 6-June-2011]. Cited on page 4.

[26] Wikipedia. IPhone (original) — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed 9-June-2011]. Cited on page 6.

[27] Zdziarski, J. *iPhone Forensics: Recovering Evidence, Personal Data, and Corporate Assets.* O'Reilly Media, 2008. Cited on pages 4, 10, 13, 16, 21, 22, and 24.

[28] ZDZIARSKI, J. Bypassing iPhone 3G[s] Encryption, July 2009. [Online; accessed 9-June-2011]. Cited on page 4.