An Open Source Toolkit for iOS Filesystem Forensics

Ahmad Cheema, Mian Iqbal, Waqas Ali

Chapter 15

AN OPEN SOURCE TOOLKIT FOR iOS FILESYSTEM FORENSICS

Ahmad Raza Cheema, Mian Muhammad Waseem Iqbal and Waqas Ali

Abstract Despite the fact that every iOS release introduces new security restrictions that must be overcome in order to recover data from iPhones, the locations where the data of interest resides are generally consistent. This paper analyzes the iOS filesystem and identifies files and directories that contain data that can aid investigations of traditional crimes involving iPhones as well as hacking and cracking attacks launched from iPhones. Additionally, best practices for minimizing the false positive rate during data carving are identified. These findings are implemented in an open source forensic investigation toolkit that operates in a forensically-sound manner.

Keywords: Mobile phone forensics, iOS, iPhone

1. Introduction

The Apple iOS operating system is a Unix-like operating system based on FreeBSD. Despite the open source roots of iOS, the operating system is locked down and accessing the core Unix functions requires an iOS device such as an iPhone to be jailbroken. An iPhone has two filesystem partitions: one partition stores iOS-specific files such as kernel images and configuration files, and the other stores user-specific settings and applications [6]. From a forensic perspective, the second partition is more important because it contains user applications and data. The call history, short messaging service (SMS) messages, contact list, emails, audio and video, and pictures taken with the built-in camera are all located in the second partition.

Since iOS is closed source and proprietary, general purpose forensic techniques and tools do not work on iOS devices. This paper analyzes the iOS filesystem and identifies files and directories that contain data

that can aid investigations of hacking and cracking attacks launched from iPhones. The paper also describes an open source forensic toolkit and forensic procedures that can extract and analyze data stored on iPhones running iOS versions 3.1.3 through 4.3. The toolkit has been tested on an iPhone 3G running iOS version 3.1.3.

2. Related Work

The iOS filesystem stores a large number of relevant files in multiple locations. The files are stored in a variety of formats (both open and proprietary); therefore, without knowledge of the location and purpose of each file, a forensic investigation is bound to fail.

Companies that market forensic software for iPhones and Android devices claim that the software can securely recover data from these devices [3]. However, because of the lack of independent verification and the closed source nature of the forensic investigation tools, the procedures used by the tools are questionable.

Schmidt [9] discusses the bruteforcing of iPhone passcodes in a safe manner without any loss of data; it is not possible to guess the passcode through the graphical interface because the data is automatically erased after ten unsuccessful attempts. Zdziarski [11] has developed a custom firmware loading and acquisition method that is forensically secure. However, the tools and techniques developed by Zdziarski are only available to the law enforcement community and are applicable to the older iOS 2.x firmware versions. Hay, *et al.* [4] have updated Zdziarski's process to work on firmware versions through 4.x; they also provide details about the locations where valuable data resides. Mallepally [7] discusses a number of challenges involved in extracting evidence from iOS devices, including novel anti-forensic techniques.

3. Implementation

Designing a forensic investigation toolkit requires care in order to ensure data integrity and that evidence is not lost. The NIST Computer Forensics Tool Testing Program for Mobile Devices [8] requires that a forensic toolkit must perform a complete data extraction and must maintain the forensic integrity of the data. The open source toolkit [5] described in this paper meets both requirements. The data integrity requirement is achieved by unmounting the iOS filesystem and then copying the two partitions using the `dd` utility via a secure shell (SSH) connection between the phone and investigator machine over Wi-Fi. Unmounting the filesystem prevents the accidental writing of data to the filesystem and, thus, maintains forensic integrity during data extraction.

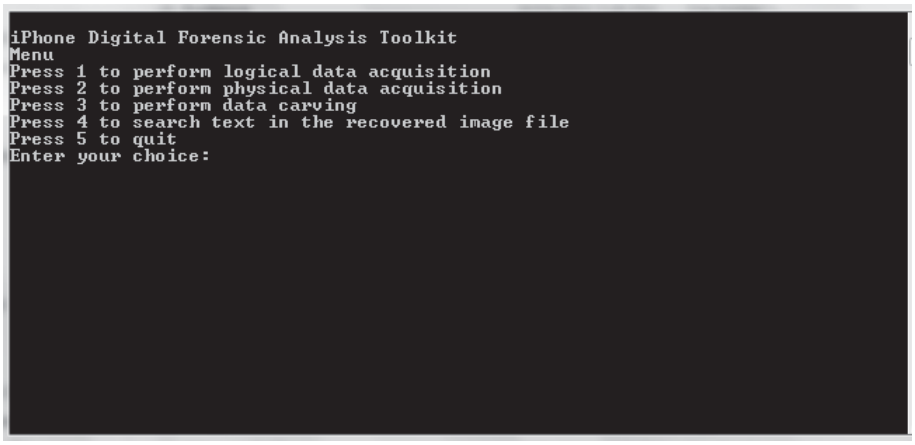

Figure 1. Screenshot of the forensic investigation toolkit.

The forensic toolkit is written in C#. Because iOS requires executables to be digitally signed, an iPhone must be jailbroken to circumvent this restriction and obtain root privileges. Root privileges can be gained using manual or automated jailbreaking methods. A manual method provides more control over the jailbreaking process. However, an automated method [1, 10] can be used because jailbreaking only changes files in the first partition. User data and other information in the second partition remain intact no matter which method is used.

After obtaining root access to the iPhone, an OpenSSH server is installed on the device. A secure shell client implemented in the forensic investigation toolkit is then used to connect to the OpenSSH server and extract data from the device to the forensic investigation machine. This process is secure because all the data is transferred through an encrypted SSH tunnel, which ensures data confidentiality and integrity

Figure 1 shows a screenshot of the forensic investigation toolkit. The toolkit menu is organized as follows:

- **Logical Acquisition:** Important files stored in the iOS filesystem are transferred from the iPhone to the forensic investigation machine. No deleted data is recovered and only the files stored in the filesystem are recovered. This type of acquisition is useful for quickly searching and retrieving evidence.

- **Physical Acquisition:** A bit-for-bit copy of the data is obtained. This type of acquisition provides the forensic investigator with a complete copy of all the data stored on the device.

Table 1. iOS filesystem evidence files.

File Name	File Location	Description
General.log	/Library/logs/ AppleSupport/	iPhone firmware information
localtime@	/private/var/db/timezone/	Local time zone configuration details
*.deb	/private/var/mobile/ Library/Backup	Downloaded application install packages
Status	/private/var/lib/dpkg/	Application installation status
Each directory	/private/var/stash/ Applications/	Install location of each application
AddressBook .sqlitedb	/private/var/mobile/ Library/AddressBook/	User contact list
Calendar.sqlite	/private/var/mobile /Library/Calendar/	User calendar data
Call_history.db	/private/var/mobile /Library/CallHistory/	Details of the last 100 calls placed, received and missed
Voicemail.db	/private/var/mobile /Library/voicemail/	Information about voicemail senders
sms.db	/private/var/mobile /Library/SMS/	Default SMS database file containing SMS messages sent and received
DraftMessage. plist	/private/var/mobile /Library/Draft/PENDING /.draft/	SMS messages written in the Messages application but not yet sent
Email	/private/var/mobile /Library/Mail/	"Protected Index" file with email information
SafariHistory .plist	/private/var/mobile /Library/Safari/	Safari browser website history information
SafeBrowsing.db	/private/var/mobile /Library/SafeBrowsing/	Websites visited using the Safari Safe Browsing feature
Cookies.plist	/private/var/mobile /Library/Cookies	Website cookies saved for the Safari browser

- **Data Carving:** After the bit-for-bit copy is obtained, the Scalpel open source forensic data carving tool is used to recover deleted data.

4. Analysis

After manually analyzing the iOS file contents, the files listed in Table 1 were identified as being relevant to iPhone investigations. The file content of interest includes address book data, SMS messages, email, images and audio/video. iPhones have been used to attack networks

Table 2. Recommended file sizes for data recovery.

File Name	Type	Size
`sms.db`	Default SMS file	1.2 MB to 2 MB
`AddressBook.sqlitedb`	Default address book	220 KB
`Notes.db`	Notes file	50 KB
`Calendar.sqlitedb`	Calendar entries	204 KB
`Call_history.db`	User call history	28 KB
`History.plist`	Safari browser history	1 MB
`Voicemail.db`	Voice mail information	100 KB

and the digital infrastructure [2]; the installed application files listed in Table 1 are also useful for investigating these crimes.

Most of the files are in the open source SQLite database format. SQLite is an efficient and lightweight database engine. However, SQLite database files lack a trailing signature and can only be identified by their file header (`53 51 4C 69 74 65 20 66 6F 72 6D 61 74 20 33 00`). This makes it difficult to recover deleted files; searching for files using only the header signature yields a large number of false positives. However, our research indicates that the false positive rate is greatly reduced if the maximum file sizes listed in Table 2 are chosen.

4.1 System Information

The `General.log` file is the first file that should be analyzed. It contains information about the iOS version installed on the device, which is important because it provides insights into the protection measures that are in place and the vulnerabilities that can be exploited in a forensic investigation. The iOS version information can also help the forensic practitioner select the appropriate tools for data recovery and analysis. The `General.log` file also stores the date when the firmware was installed; this gives an estimate of how long the device has been in use.

Another important file is `localtime`, which contains local time settings data that is useful for developing an accurate timeline of user activity involving an iPhone.

4.2 Installed Applications

The `/private/var/mobile/Library/Backup` directory contains multiple `.deb` files. These files correspond to Debian package management system install packages; their presence indicates that they were downloaded by the user in preparation for installation on the iPhone. Even if an application is subsequently removed from the device, its down-

loaded package persists until the maximum archive size specified in the `apt.conf` file is reached. Of particular interest in investigations are `.deb` files corresponding to hacking and cracking applications.

Together with the `.deb` archive files, the status file can be used to establish that a user not only downloaded applications but also installed them. The status file contains the status of every package ever installed on the iPhone, including applications downloaded and installed from Cydia repositories

Each installed application has a sub-directory in the `/private/var/stash/Applications` directory. This is useful when a user has deliberately hidden applications from the normal display. For example, applications are available for hiding applications from the SpringBoard GUI until a particular key combination or password is entered. This protection can be circumvented by examining the contents of the `/private/var/stash/Applications` directory.

4.3 User Data

The most important files for investigating traditional crimes are located in the `/private/var/mobile/Library/` directory. This directory and its sub-directories contain the address book, calendar, call history, SMS messages, email and web browser history.

The `AddressBook.sqlitedb` file contains contact information and numbers stored in the iPhone. Two tables, ABPerson and ABMultiValue, are especially important. ABPerson contains contact names while the ABMultiValue table has the corresponding contact numbers.

The `Calendar.sqlitedb` file stores user calendar entries. Another important file is `Call_history.db`, which contains the last 100 calls dialed, received and missed. The `Voicemail.db` file stores the sent and received voicemail messages.

The message table in the `sms.db` file contains the sent and received SMS messages. Data extracted from this table must be correlated with the data stored in the ABPerson and ABMultiValue tables to determine the contact information of the person who sent an SMS message. Relating the data with `AddressBook.sqlitedb` requires the execution of the following command line query:

```
ATTACH DATABASE "AddressBook.sqlitedb" AS AB;
ATTACH DATABASE "sms.db" AS sms;
SELECT First,Last FROM AB.ABPerson WHERE rowid=(SELECT
record_id FROM AB.ABMultiValue WHERE value=(SELECT
address FROM sms.message WHERE text LIKE \%\%));
```

The `DraftMessage.draft` file contains SMS messages that have not yet been sent. By default, iOS does not have a draft saving feature. However, messages written using the Messages application, but that have not yet been sent are saved in this file.

Email messages are stored in the "Protected Index" file. This file has no file extension, but opening it in a hex editor and examining the header reveals that it is a SQLite database file. Two tables, messages and messages_data, contain important forensic artifacts. User browsing history artifacts are found in the `History.plist`, `SafeBrowsing.db` and `Cookies.plist` files.

5. Evaluation

This section presents two use cases: one involving an iPhone seized in a traditional criminal investigation and the other involving an iPhone used in a hacking incident. The two cases highlight how the results of this research can support digital forensic investigations.

- **Traditional Crime Use Case:** A phone is often one of the most important evidence containers in traditional crimes such as theft and murder. Traditional criminal cases typically focus on the contact list, SMS messages, email, images, audio/video and call history. Recovering data from an iPhone can be difficult if proper tools and techniques are not employed. The open source tool described in this paper was tested on an iPhone 3G with 8 GB internal storage running firmware version 3.1.3.

  The first task in a forensic investigation is to jailbreak the iPhone to obtain root access and bypass the security restrictions. This should be done using offline jailbreak tools such as JailbreakMe [1], which can jailbreak iPhones up to firmware version 4.3. Jailbreaking only alters the data in the first partition; no data (including user data) residing in the second partition is modified or lost.

  After the phone is jailbroken, the next step is to use the Cydia repository to install OpenSSH on the device. Then, the open source toolkit [5] described in this paper is used to download the files in the `/private/var/mobile/Library` directory in order to perform offline analysis. The most important data is the call history, contact list, SMS messages and email, all of which are stored in SQLite files that can be analyzed with a SQLite browser (e.g., the open source SQLite Browser) or hex dump software. Audio and video files are also recovered from their sub-directories and analyzed.

Figure 2. Applications installed on an iPhone.

- **Hacking Incident Use Case:** In the case of an iPhone involved in a hacking or cracking case, it is necessary to the analyze specific application files used in the illegal activities as well as the configuration and log parameters. Commonly-used hacking tools such as Nmap, Metasploit and Aircrack-ng can only be installed if the device has already been jailbroken. The problem is that when an iPhone is jailbroken to conduct an investigation, important log files and application files related to the incident are modified because they are also stored in the first logical partition that is affected by a jailbreaking process. Since an iPhone involved in a hacking or cracking case has already been broken to load the attack tools, there is no need to jailbreak the device and the forensic integrity of the device is not impacted.

  The most important files in hacking and cracking investigations are files located in the installed application directory `/private/var/stash/Applications` and the `.bash_history` file located in the `root` directory. The installed application directory lists the applications installed on an iPhone (Figure 2) while the `.bash_history` file lists all the commands executed via the terminal.

6. Conclusions

The Apple iPhone is a modern smartphone that provides sophisticated computing, communications and user functionality. The proprietary and closed nature of the iPhone hardware and software greatly complicates forensic recovery and analysis. The analysis of the iOS filesystem has identified files and directories that contain evidentiary data of interest

to traditional criminal investigations as well as investigations of hacking and cracking attacks launched from an iPhone. The open source forensic toolkit described in this paper is specifically designed to extract and analyze evidentiary data stored on iPhones running iOS versions 3.1.3 through 4.3.

References

[1] comex, JailbreakMe (`www.jailbreakme.com`), 2011.

[2] D. Compton, Own with an iPhone (`www.youtube.com/watch?v=zBm1UXmgz1k`), 2010.

[3] ElcomSoft, ElcomSoft Phone Password Breaker, Moscow, Russia (`www.elcomsoft.com/eppb.html`), 2014.

[4] A. Hay, D. Krill, B. Kuhar and G. Peterson, Evaluating digital forensic options for the Apple iPad, in *Advances in Digital Forensics VII*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 257–274, 2011.

[5] W. Iqbal, Open source toolkit for iPhone filesystems file analysis (`sites.google.com/a/mcs.edu.pk/open-source-toolkit-for-iphone-file-system-files-analysis/open-source-toolkit-for-iphone-file-systems-file-analysis`), 2013.

[6] R. Kubasiak, S. Morrissey, W. Barr, J. Brown, M. Caceres, M. Chasman and J. Cornell, *Mac OS X, iPod and iPhone Forensic Analysis*, Syngress Publishing, Burlington, Massachusetts, 2009.

[7] R. Mallepally, Implementation of Applications to Improve iPhone Forensic Analysis and Integrity of Evidence, M.S. Thesis, Department of Computing Sciences, Texas A&M University – Corpus Christi, Corpus Christi, Texas, 2011.

[8] National Institute of Standards and Technology, Mobile Devices, Computer Forensics Tool Testing Program, Gaithersburg, Maryland (`www.cftt.nist.gov/mobile_devices.htm`).

[9] J. Schmidt, iOpener: How safe is your iPhone data? *The H Security*, Heise Media UK, London, United Kingdom (`www.h-online.com/security/features/iOpener-How-safe-is-your-iPhone-data-1266713.html`), July 4, 2011.

[10] J. Sigwald, Analysis of the jailbreakme v3 font exploit, Sogeti ESEC Lab (`esec-lab.sogeti.com/post/Analysis-of-the-jailbreakme-v3-font-exploit`), 2011.

[11] J. Zdziarski, *iPhone Forensics*, O'Reilly Media, Sebastopol, California, 2008.