OPEN ACCESS
Indo-JC
Indonesia Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 4, Issue. 2, Sept 2019. pp. 205-218
doi:10.21108/indojc.2019.4.2.349

Digital Forensic Analysis on iDevice: Jailbreak iOS 12.1.1 as a Case Study

Amin Aenurahman Ali [1], Niken Dwi Wahyu Cahyani [2], Erwid Musthofa Jadied [3]

[1,2,3] *Informatics, School of Computing, Telkom University*
Jl. Telekomunikasi No. 1 Terusan Buah Batu, Bandung, Indonesia

[1] altear@student.telkomuniversity.ac.id
[2] nikencahyani@telkomuniversity.ac.id
[3] jadied@telkomuniversity.ac.id

Abstract

Jailbreak has an issue in data alteration, as it modifies file(s) in the device to allow user to extract more data than without jailbreaking. This issue raises controversy of the use of jailbreaking in digital forensic investigation, as data integrity is a prominent requirement in a court proceeding. This study aims to analyze the process of jailbreak, what is actually done by the jailbreak code in a device, and what data is actually modified by the jailbreak code. By using the latest version of iOS system, this study uses the voucher_swap exploit as a representation of semi-tethered jailbreaking method to investigate the effects of jailbreak on data integrity on a idevice. The investigation is conducted based on to what extent data can be extracted from the jailbreak device, hash value comparison of the data, and source code analysis to scrutinize the effect of jailbreak to the system and user data inside the device. Results of this study suggest that jailbreak is acceptable to prepare idevice in digital forensic investigations to acquire more data, as it maintains the integrity of user data. These results may help forensic communities, especially investigators in their decision about the acceptability of jailbreaking in idevide forensic investigations.

Keywords: digital forensics, iOS, iPhone, jailbreak, root privilege

Abstrak

Jailbreak memiliki masalah dalam perubahan data, karena memodifikasi file di perangkat untuk memungkinkan pengguna mengekstrak lebih banyak data daripada tanpa jailbreak. Masalah ini menimbulkan kontroversi penggunaan jailbreak dalam investigasi digital forensik, karena integritas data merupakan persyaratan utama dalam proses pengadilan. Penelitian ini bertujuan untuk menganalisis proses jailbreak, apa yang sebenarnya dilakukan oleh kode jailbreak di perangkat, dan data apa yang sebenarnya dimodifikasi oleh kode jailbreak. Dengan menggunakan versi terbaru sistem iOS, penelitian ini menggunakan exploit voucher_swap sebagai representasi dari metode jailbreak semi-tethered untuk menyelidiki efek jailbreak pada integritas data pada sebuah idevice. Penyelidikan dilakukan berdasarkan sejauh mana data dapat diekstraksi dari perangkat jailbreak, perbandingan nilai hash data, dan analisis kode untuk meneliti efek jailbreak terhadap sistem dan data pengguna di dalam perangkat. Hasil penelitian ini menunjukkan bahwa jailbreak dapat diterima untuk mempersiapkan idevice dalam investigasi forensik digital untuk memperoleh lebih banyak data, karena menjaga integritas data pengguna. Hasil ini dapat membantu komunitas forensik, khususnya para penyelidik dalam keputusan mereka tentang penerimaan terhadap penjatuhan hukuman dalam penyelidikan forensik pada idevide.

Kata Kunci: digital forensik, iOS, iPhone, jailbreak, root privilege

Received on August 2019. Accepted on Sept 2019

I. Introduction

IOS was first introduced in 2007 on an iPhone device [1]. The iOS was developed by the Apple Machintosh Team, which is officially called the iPhone OS. Then it was changed to iOS in 2010. iOS can be used on iPhone, iPad and iPod Touch devices. iOS has been going on for eleven years and has undergone changes and increased security since it was first launched [2]. At the time of writing this paper, iOS 12.3 is the latest stable version that can be used.

Jailbreak has an issue in data alteration. It somehow needs to modify files in the device in order to extract more data. This file modification raises concern of data integrity as the prominent consideration in forensics examination. We need to maintain the data integrity that is required by the court proceeding, by conducting data extraction in a forensically sound manner. However, as in contrast, one proprietary forensic tool states that jailbreaking is necessary to perform physical acquisition stage in order to do file system extraction and keychain decryption, for 64-Bit iOS [3]. Without jailbreaking, the tools can only support logical acquisition that limit data extraction only on idevice information, iTunes format backup, list of installed apps, media and shared files [4]. The pros and cons of data integrity issue in a jailbreaking device makes the use of this method in forensic examination is needed to be examined further, as it may influence the acceptability of digital data in a court of law.

However, despite of this importance issue of data modification in jailbreak, there are a few studies that investigate to what extent jailbreaking modify the data, either system data or user data. Previous investigation on the controversy of jailbreaking, in 2015, clarifies that this procedure will not change the internal digital evidences of iPhone [5]. As iOS operating system and jailbreaking tools developed, we need to keep up to date to analyze the process of jailbreak, what is actually done by the jailbreak code in a device, and what data is actually modified that may raise concern in data integrity.

Therefore, this study aims to scrutinize the jailbreak impact to the latest version of iOS device in data integrity, by conducting a digital forensic analysis particularly in iOS device as it is known as a device that has taken care of users privacy and security on the top level of its architecture that may complicate the data extraction process. Results of this research can be used as a basis by investigator, to decide whether the jailbreak can be accepted or not by the forensic community to conduct an investigation on iOS.

In section I of paper contains the background of digital forensic on the idevice jailbreak. In section II explain the jailbreak process on iOS and literature review. In section III explain the design of a digital forensic test is done on the idevice, including in the form of flow testing as well as equipment needed for the test. Then in section IV described in detail the results and discussion. Finally, in section V the conclusions were given.

II. Literature Review

A. Mobile Forensic

Digital forensics is a branch of forensic science covering the recovery and investigation of the material found in digital devices, usually often related to computer crimes. The main goal of digital forensics is the extraction of suspected files from target devices that can be defined as digital proof.

The digital science of forensic uses the tools, techniques and method that scientifically proven, which can be used to acquire and analyze digital evidence. There is a need for law enforcement agencies and governments, even private organizations to invest the advancement and development of digital forensic technology itself.

According to the Association of Chief Police Officer (ACPO) there are four basic principles used in digital forensics that are commonly adopted by forensic communities, namely follows [6]:

- A law enforcement agency and/or its officers are prohibited from altering digital data stored in an electronic storage medium that will subsequently be carried over and liable in court.

- For someone who feels the need to access digital data stored in the media of a proof of storage, then that person must be completely clear in their competence and can explain the relevance and implications of the actions he or she inspection and analysis of the evidence.
- There should be a technical and practical record of the measures applied to the storage media of evidence in the event of examination and analysis, so that when the evidence is examined by a third party then the parties should these three will result in the same result as the previous forensic investigator/analyst has done.
- A person who is responsible for the case or examination and analysis of electronic evidence should be able to be aware that the process that takes place in accordance with the laws of the law and the basic principles (i.e. basic principle number 1, 2 and 3) can be properly communicated.

Then there are stages and procedures that are done in digital forensics. The first is preparation. This stage is useful for setting up everything that will be needed during the investigation process. The second stage is the acquisition stage. At this stage digital evidence that serves as a storage media will be done imaging process, which is the process of doubling the contents of the storage media used and will generate the image file. However, the computer/assistive device used in the imaging process must be equipped with write protect. This Write protect is intended to maintain the integrity of the content of evidence. In digital forensic, data integrity enforcement is done by comparing the hash values of the original data and duplicating data. The use of hash values for data integrity testing has been generally accepted in the forensic community, as asserted by The Scientific Working Group on Digital Evidence (SWGDE) (2006, p.3) "Digital Evidence submitted for examination should be maintained in such a way that the integrity of the data is preserved. The commonly accepted method to achieve this is to use a hashing function." [7]. Next is the examination phase. The Image file obtained at the acquisition stage will be examined comprehensively with the intention of obtaining digital data in accordance with the investigation. Last is the reporting phase. This report contains data that is derived from the investigation and analysis process in accordance with the investigations.

B. iOS

1) iOS Boot Process: Here is the booting process on iOS and the various jailbreak that can be done on iOS

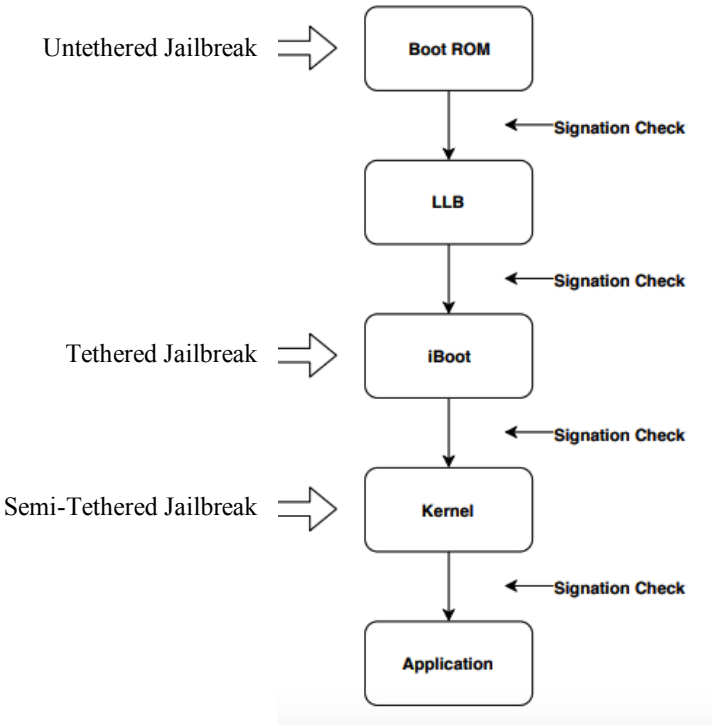


Fig. 1. Booting process on iOS and various types of jailbreak attacks [8]

At the Fig. 1, first time the iPhone is turned on, it will read system startup code (Boot ROM) where the CPU will only read the read-only area. This Boot ROM will encrypt the next Low-Level Bootloader (LLB) startup stage containing the root certificate of the boot Integrity test. The next step will be detection on the iOS system based on the command code on iBoot and kernel before it can finally run an application. After all stages, the system will be tested to ensure that the system is ready for use [9].

2) Architecture of iOS: iOS has a resemblance to macOS, which is equally Unix-based, As iOS development is based directly on OS X development. The APIs used on iOS also use Cocoa Touch so that the app can interact with OS X, as described in the following image.

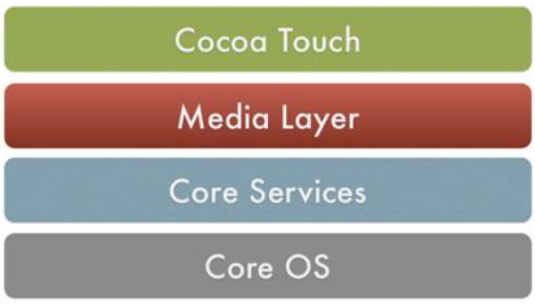


Fig. 2. IOS architecture is a layered architecture that communicates with each other [10]

The iOS architecture in Fig. 2 is an architecture that has layers, where the topmost layer serves as an intermediary between the hardware and the software used. The application does not directly interact with the

hardware, thus communicating through a set of well-defined system interfaces. This interface makes it easy to write applications that work constantly on a device.

System components of the Core OS Layer includes the operating system kernel and the kernel is the foundation on which the entire iOS platform is built and provides the low level interface to the underlying hardware. Amongst other things, the kernel is responsible for memory allocation, process lifecycle management, input/output, inter-process communication, thread management, low level networking, file system access and thread management [11].

3) Kernel of iOS: Based on the perspective file system, iOS is very similar to OS X. The iOS kernel already exists in packages with kernelcaches extension (/System/Library/Caches/com.apple.kernelcaches) and encrypted using IMG3 [12]. Table I display the detail.

TABLE I
KERNEL DIRECTORY ON OS X AND IOS [12]

OS	System/Library/Caches/…	Contains
OS X	com.apple.kext.caches/Startup	Mach-O binary
iOS	com.apple.kernelcaches/kernelcache	Kernelcache in IMG3 encrypted form

To see similarities can be done by using Apps Terminal with command "uname -a". Uname -a is used to print the name, version and other spesifications about the OS and the system.

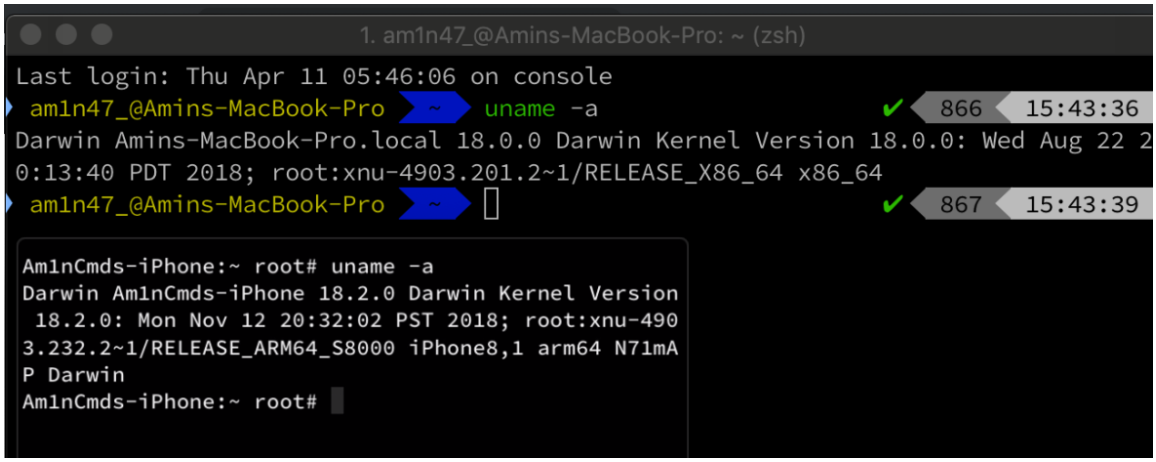


Fig. 3. Usage "uname -a" command to display details

Fig. 3 showing results of checking on OS X and iPhone using the "uname-a" command show similar results where the XNU (Apple-developed optical system kernel) on OS X and iPhone shows the same version of XNU-4903.

4) Jailbreak on iOS: Jailbreaking is the process to bypass restrictions on iOS to install other applications and tweaks that are not allowed by Apple's party. The main purpose of jailbreaking is to get a superuser right that allows root access to system files, so users can perform the activity without any limitations by Apple. Like installing an app that doesn't exist in the App Store.

Previous research about forensic on iOS jailbreak has explained by Jonathan Zdziarski [13], using iPhone 4 iOS 4.3.3 where the architecture is 32-bits with the HFS+ format file system. However, that product has been discontinue.

Continuing the research, this paper describes digital forensic investigations conducted on the iPhone 6s with iOS 12.1.1 that can be jailbroken. This iPhone 6s uses a 64-bit architecture as well as a new scheme of the APFS format file system [14]. Jailbreak on iOS 12.1.1 is done using the Unc0ver tool which is a semi-tethered

jailbreak. This type needs to start the jailbreak process every time the idevice is turned off or restarted, because exploits that can be used now only reach the kernel stage, unlike used on iOS 4.3.3 using the redsn0w tool that has exploits up to the Boot ROM stage.

Jailbreaking modern versions of iOS is an extremely complex process exploiting multiple vulnerabilities in various parts of the OS to defeat the systems' security measures [15]. In general terms, a jailbreak performs the following steps:

- Sandbox Escape: during this stem, the jailbreak tool utilizes an exploit allowing it to access components it does not have the permissions to.
- Privilege Escalation: the jailbreak gains elevated privileges allowing it to access protected resources (e.g. mount the root file system, patch the kernel, inject code etc.)
- KPP Bypass: disables or works around the code signing check, which allows modifications to the file system without making the device unbootable, causing a bootloop or random reboots.

While getting more complicated, modern jailbreak tools are safer to use. Starting with iOS 11, all jailbreaks are utilizing the same installation procedure. A failed jailbreak does not cause system instability, and does not required reinstalling iOS in order to perform another attempt.

Jailbreaking on iOS violates Apple's EULA and eliminates the warranty on its device. However, in 2010, the Electronic Frontier Foundation (EFF) was able to obtain certain exceptions that were converted into the Digital Millenium Copyright Act (DMCA). It keeps the jailbreak community safe from lawsuits [16].

There are 3 known types of jailbreak, namely:

- Tethered Jailbreak

  Tethered jailbreak is a jailbreak that requires a computer-assisted tool. The idevice will be connected to the computer using a lightning cable. This tethered jailbreak is temporary. Due to its temporary nature, the jailbreak can only be used in a single boot. A tethered jailbreak will exploit the iBoot stage.
- Untethered Jailbreak

  Untethered jailbreak is a jailbreak that does not need a computer unless the first process jailbreak. Tethered jailbreak is permanent, because every time the idevice restarts no longer need to be connected to the computer to activate jailbreak, including when running out of battery. This untethered jailbreak uses the exploit on the Boot ROM as well as installing additional exspoitation on the root file system.
- Semi-Tethered

  Semi-Tethered jailbreak is a process of jailbreak that can be done with or without using a computer. This type of Jailbreak usually performs attacks on the kernel stage. This type of Jailbreak is done after the iPhone is live and start running the application and should be done every time after the iPhone restarts/turns off.

III. Research Method

A. Testing Flow

The test in this study follows the testing flow as described in Fig. 4 below. The scenario starts with the acquisition stage on iOS jailed to collect evidence, after obtaining evidence data, the next process is to preserve it to take care of the same thing or prevent it from being damaged. Preservation data is then separated from the original data for later analysis. Then the idevice will be jailbroken to make a comparison to the jailed idevice. On jailbroken devices, the acquisition and preservation process is done again to get data for analysis. Analysis is based on the files obtained at the preservation stage, to determine the extent to which data can be extracted using a jailbreak and the effect of jailbreak on the integrity of a data.

To strengthen the evidence for analysis, we also use the reverse engineering technique in the source code exploit used in the jailbreak tools. The source code exploit was obtained from https://bugs.chromium.org/p/

project-zero/issues/detail?id=1731 to find out the changes made from jailbreak against the iOS system, until finally the results of the analysis are obtained and can be presented.

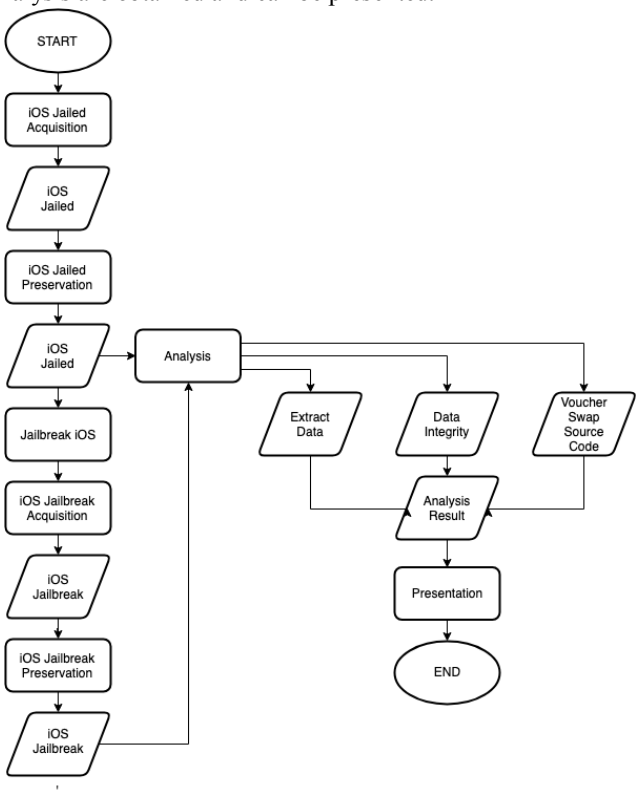


Fig. 4. Digital forensic testing flow on idevice

B. Tools

Tests were carried out on the iPhone 6s with the specifications described in Table II. In this test also needed several devices that would support the testing process. The device is divided into two, namely software (software) and hardware (hardware) with the specifications described in Table III and Table IV.

Experimental dataset in Table V is collected and downloaded from https://cdn.online-convert.com/example-file/, two large text file downloaded and extracted from https://datasets.imdbws.com and one large video files downloaded from http://podcasts.apple.com/apple_keynotes_1080p/2018/october2018_1080.m4v have been copied to the iPhone. In one file type it consists of various formats with different amounts.

TABLE II
IPHONE SPESIFICATION

No	Specification	Detail
1	Name	iPhone 6s
2	Identifier	iPhone8,1 (A1633)
3	Platform	s8000
4	BoardConfig	n71map
5	Disk Type	TLC
6	OS Version	12.1.1 Build 16C50
7	Firmware	iPhone_4.7_12.1.1_16C50_Restore.ipsw
8	Card Slot	No
9	Internal	128GB
10	RAM	2GB

TABLE III
SOFTWARE SPESIFICATION

No	Software	Detail
1	OS	macOS Mojave 10.14.0
2	Tools	- iTunes - Xcode - Jtool2

TABLE IV
HARDWARE SPESIFICATION

No	Hardware	Keterangan
1	Type	Dell Inspiron 14 7447 Pandora
2	CPU	Intel i7-4720HQ (8) @ 2.60GHz
3	RAM	8 GB 1600 MHz DDR3
4	IGPU	Intel HD Graphics 4600
5	GPU Discrette	Nvidia GTX950M 4GB DDR3 (disabled)
6	Storage	SSD Samsung 850 Pro 256GB + HDD Toshiba 1TB
7	Cable	Lighning to USB

TABLE V
EXPERIMENTAL DATASET

No	Type	Detail
1	Audio	6 file (.aac, .amr, .flac, .m4a, .m4p dan .mp3)
2	Dokumen	49 file (.xls, xlsx, .csv, .ppt, .pptx, .doc, .docx, .pdf, .rtf dan .txt)
3	File	8 file (.7z, .rar, .zip dan .bin)
4	Gambar	14 file (.gif, .heic, .jpg, .jpeg, .png dan .tiff)
5	Video	21 file (.3gp, .avi, .flv, .h264, .m4v, .mp4, .mpg, .mpeg, .swf, .webm and .wmv)
6	Large Files	2 text file (title.basics.tsv and name.basics.tsv) over 500MB and 1 video files (october2018_1080.m4v) over 5GB

IV. RESULTS AND DISCUSSION

A. Data Extraction

The first test was carried out on a jailed iPhone condition to check the extent to which data can be extracted. Then the same thing is done also when the condition of the iPhone has been jailbroken. The results show the different location of the directory where the jailed iPhone shows that the File System (User) is in the root directory (/) while in the iPhone jailbreak condition, the File System (User) is in the /private/var/mobile/Media directory.

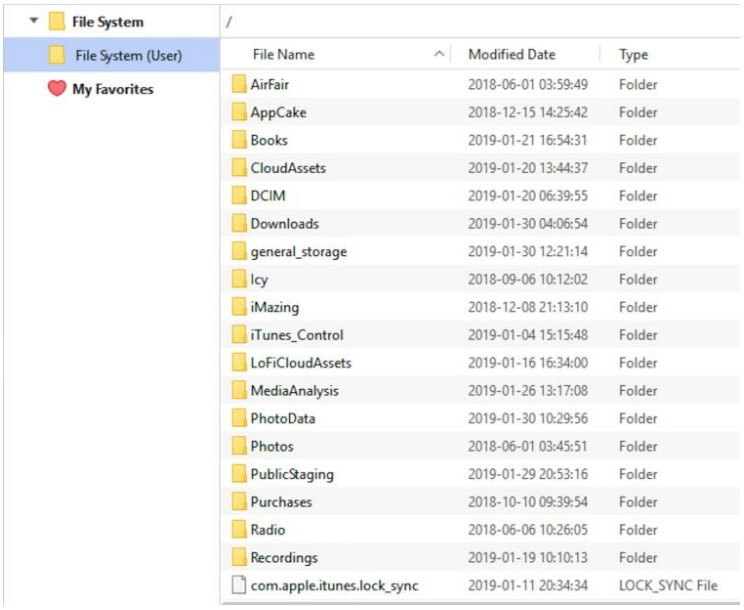


Fig. 5. Extraction data result from iPhone jailed

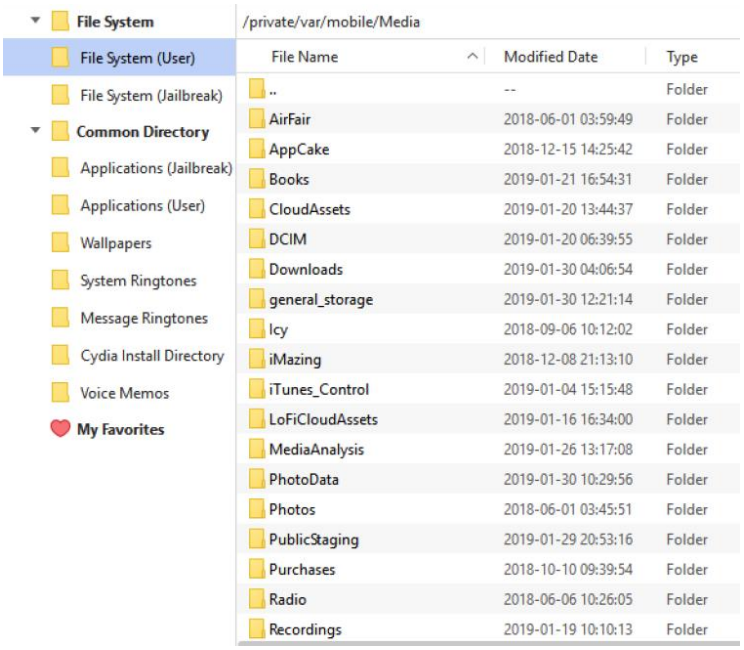


Fig. 6. Extraction data result from iPhone jailbreak

Based on Fig. 5 and Fig. 6 we can investigate the directory as detailed below:

- File system (User) located in /private/var/mobile/Media is a directory for user saving their data
- File system (Jailbreak) located in / (root) directory
- Application system like Appstore.app, Camera.app, iCloud.app and some jailbreak application is located in /Application directory
- Application downloaded by user from Appstore or any other Sign Services are located in /var/mobile/Containers/Data/Application
- Wallpaper default directory from iOS system is located in /Library/Wallpaper

- Message ringtones default directory from iOS system are located in /Library/Ringtones
- Voice memo is located in /private/var/mobile/Media/Recordings because its user Media

B. Data Integrity

To test whether jailbreak has an effect on data integrity, we have done it before and after the iphone was jailbroken. The data used are experimental datasets mentioned in Table VI. Here is a summary:

TABLE VI
Comparison Result by Hashing Dataset

Audio File Name	**SHA256 Jailed**	**SHA256 Jailbreak**	**Match**
example.m4a	f7aa22884785dbb32e46501504d5dc8417 d96f70ec752e64d91100ebb58d9501	f7aa22884785dbb32e46501504d5dc8417 d96f70ec752e64d91100ebb58d9501	Yes
example.mp3	3c2298abb4c5b58c02826bbfe49a7ff7990 75b7fbcfafe0e10a1812f4bd69f31	3c2298abb4c5b58c02826bbfe49a7ff7990 75b7fbcfafe0e10a1812f4bd69f31	Yes
Document File Name	**SHA256 Jailed**	**SHA256 Jailbreak**	**Match**
example.docx	dbd23bcdd0c41585d1a232d79ffc2cba0e8 f181fbeddfd32d79f985ae7ec7c49	dbd23bcdd0c41585d1a232d79ffc2cba0e8 f181fbeddfd32d79f985ae7ec7c49	Yes
example.pdf	c7ff6c55e1f2b3fb6e364e4e4e20b325610 b041593f160fab04eebc61ae122ea	c7ff6c55e1f2b3fb6e364e4e4e20b325610 b041593f160fab04eebc61ae122ea	Yes
File Name	**SHA256 Jailed**	**SHA256 Jailbreak**	**Match**
example.rar	e39c02ed8bed5488fcb7f73255e5fee06cb a55ad808f6a24881392748bd309d1	e39c02ed8bed5488fcb7f73255e5fee06cb a55ad808f6a24881392748bd309d1	Yes
example.zip	e939e93c7941cb4f88840d591bd4226148 3364cbca5e5124da88587650478f3f	e939e93c7941cb4f88840d591bd4226148 3364cbca5e5124da88587650478f3f	Yes
Image File Name	**SHA256 Jailed**	**SHA256 Jailbreak**	**Match**
example.jpeg	0f46080d81aecb235f999308d19513ef823 71658425f8ab63b2fa5d2ae521e13	0f46080d81aecb235f999308d19513ef823 71658425f8ab63b2fa5d2ae521e13	Yes
example.png	e7fae8acb0d038f604f6a39963c5f5915d6 13963136f485716ed97a50df422e0	e7fae8acb0d038f604f6a39963c5f5915d6 13963136f485716ed97a50df422e0	Yes
Video File Name	**SHA256 Jailed**	**SHA256 Jailbreak**	**Match**
example.flv	e930fea64610173fabd68d91339a84f3857 77b6da698bd2cffc14ba3b0f235bc	e930fea64610173fabd68d91339a84f3857 77b6da698bd2cffc14ba3b0f235bc	Yes
example.mp4	e39733a03721009a154e595c5c1ee068b5 cef5c05a217b07c80a06b54a29eac2	e39733a03721009a154e595c5c1ee068b5 cef5c05a217b07c80a06b54a29eac2	Yes
Large File Name	**SHA256 Jailed**	**SHA256 Jailbreak**	**Match**
title.basics.tsv	6eb74d5ccb6c823ba0e3ecdd5425d962aa 8df02dd238b8fbcc68b8e5498dca8b	6eb74d5ccb6c823ba0e3ecdd5425d962aa 8df02dd238b8fbcc68b8e5498dca8b	Yes
name.basics.tsv	ceb2cf9053fa7625e8c89b60604123ef779 ebd3c4c34b151bc6a8325cec4a816	ceb2cf9053fa7625e8c89b60604123ef779 ebd3c4c34b151bc6a8325cec4a816	Yes
october2018_10 80.m4v	da697f4354557f4fa8e47a42b20faa64904 a75cae058b75f58e774eb8c9d9b78	da697f4354557f4fa8e47a42b20faa64904 a75cae058b75f58e774eb8c9d9b78	Yes

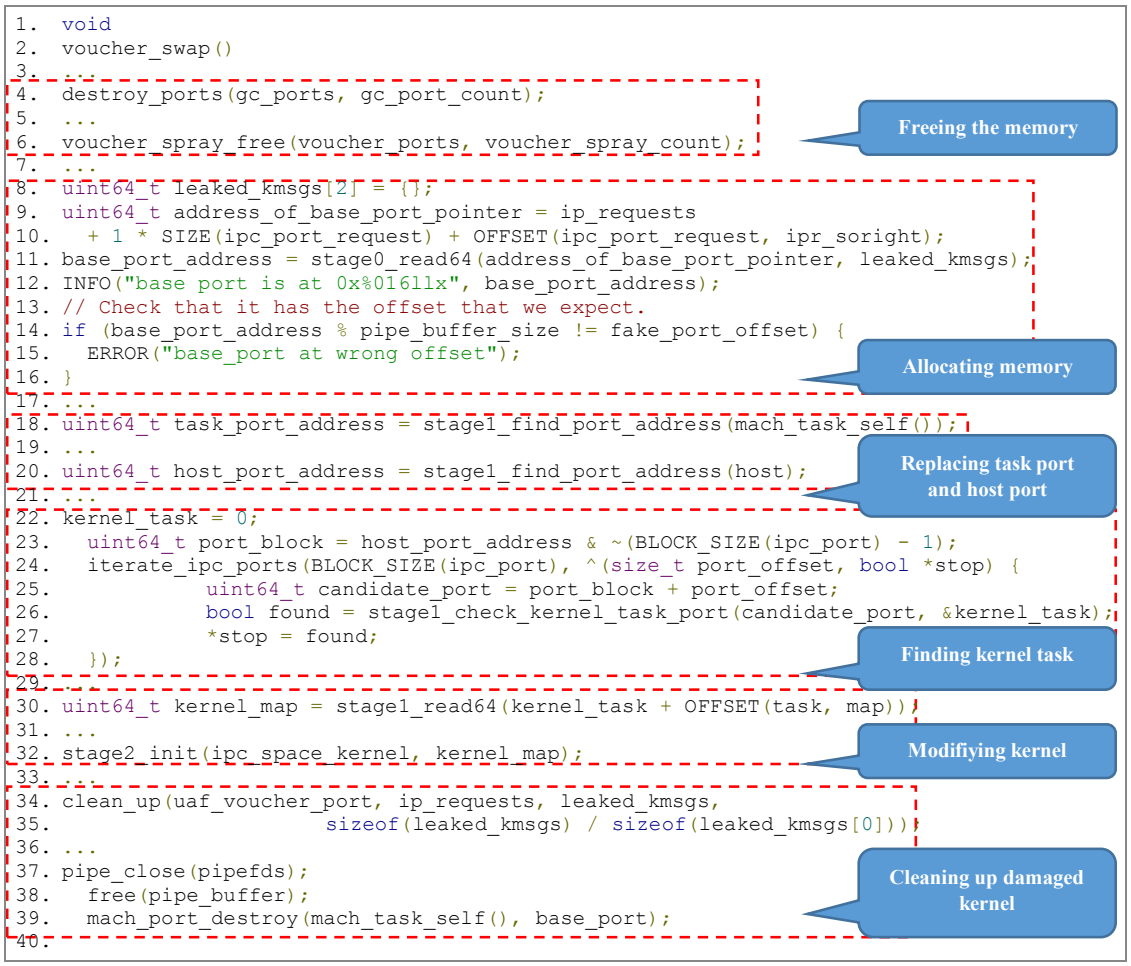
For more complete result can check https://is.gd/HashingDataset.

From the test results it appears that the jailbreak does not affect the integrity of a data, it is evidenced by the results of hashing on each of the data listed in Table V. This result is amplified by the analysis of exploit

voucher_swap which aims to override the original kernel_task so that the user has a higher previllege and has read write access up to the root directory.

C. Voucher Swap Exploit

After investigating directory, analysis continue uses by exploit which is run on Xcode. Main exploit logic is in the file voucher_swap/voucher_swap.c, in the function voucher_swap(). There are several stages until finally the kernel can read / write via kernel_task_port. We have summarized these stages as following code:



After asking with the exploit developer, we get the conclusion that original exploit itself does not modify user data [17]. On lines 4-6 aims to free memory. Continue by calling system calls to allocate memory (line 8-16) in a certain pattern in the kernel. Then on line 18-20 explain to get the task_port and host address to be replaced with fake kernel_task. On lines 22-28 do repetition to find the kernel_task address. Uses a use-after-free vulnerability to modify memory while it's being used by the kernel (line 30-32), eventually resulting in the ability to read and write arbitrary kernel memory. The last step, we clean fake kernel_task and some damaged kernel state (line 34-39), so it doesn't cause the kernel panic when it used. After this is achieved, it is possible to modify certain parts of kernel memory to escape the sandbox, at which point it can access any file. Doing it this way should fully preserve the integrity of user data (except, of course, for the fact that installed the voucher_swap app on the idevice).

As proof of the source code, we also dumped kernelcache under conditions before and after the jailbreak by using Jtool2. The results of the dumping kernel are stored as txt, then compare by hashing it. Here is hashing result of the kernelcache displayed on Table VII:

TABLE VII
Comparison Result by Hashing kernelcache

File Name	SHA256 Jailed	SHA256 Jailbreak	Match
kernelcache.txt	509ce3f769e462b3f91446b60c880c345 1dcb7aabe32e9b8f417ddeb9afa1de7	fb84107ee60f9445d42788bd5b8b5290a a73bb19f60bf2c2204d9cbe2621332d	No

For more detailed results, it can be accessed at https://github.com/Am1nCmd/Final-Task in Dumped Kernel directotry.

The following Fig. 7 shows the exploit position on iOS architecture. Here, the exploit as a jailbreak code will alter system data only in the kernel level. The green color refers to the original file, the red color refers to exploit voucher_swap works by creating a fake kernel_task and replacing the original kernel_task in kernelcache, while the grey color refers to the file is untouched.

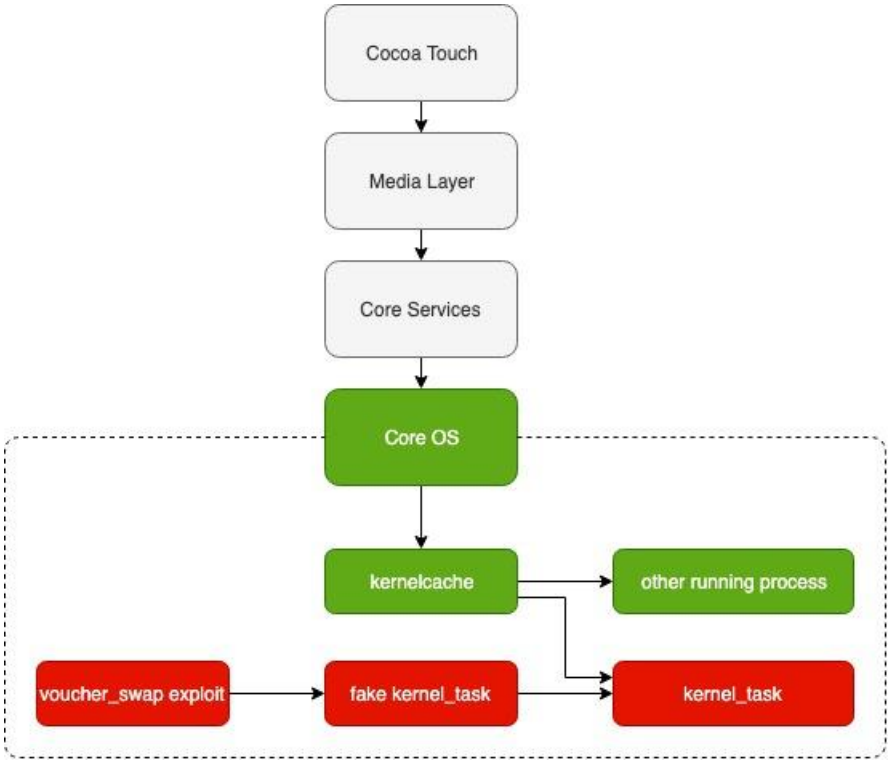


Fig. 7. Voucher_swap exploit work by modifying original kernel_task

V. Conclusion

In this study, digital forensic analysis on the latest idevice, i.e. iOS 12.1.1 is conducted to examine the process of jailbreak and its impact on data integrity. By using the unc0ver tool with voucher_swap exploit, the analysis of this semi-tethered type jailbreak has been used to compare the difference in the system and user data, before and after the jailbreaking process. It is shown that even though the device has been jailbroken, the user data in

the Media directory are not altered. Their data integrity are remain unchanged, they are shown by their match hash value before and after being jailbroken. This means that no prepared-user data has changed after jailbreaking. Nevertheless, data alteration in the system is inevitable. By analyzing the source code of the exploit used, i.e. the voucher_swap as the jailbreak code, it changes the system data by making a fake kernel_task. This fake kernel_task replaces the original kernel_task that is running in memory, to allow root privilege for data extraction. Results of this study suggest that jailbreak is acceptable to prepare idevice in digital forensic investigation to acquire more data while maintaining user data integrity.

REFERENCES

[1] S. D. Natalie Kerris, "Apple Reinvents the Phone with iPhone," Apple, 9 January 2007. [Online]. Available: https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/. [Accessed 5 November 2018].

[2] Apple, "About the security content of iOS 12.1.1," Apple, April 03 2019. [Online]. Available: About the security content of iOS 12.1.1. [Accessed 20 May 2019].

[3] Elcomsoft, "Elcomsoft iOS Forensic Toolkit," Elcomsoft, [Online]. Available: https://www.elcomsoft.com/eift.html. [Accessed 12 November 2018].

[4] D. R. S. Priyank Parmar, "Logical acquisition of iPhone without Jail Breaking," *IJSRST,* vol. 4, no. 9, pp. 2-3, 2018.

[5] K.-C. T. Y.-C. T. S.-J. W. Ya-Ting Chang, "Jailbroken iPhone Forensics for the Investigations and Controversy to Digital Evidence," *Journal of Computers,* vol. 26, pp. 21-23, 2015.

[6] A. o. C. P. Officers, "ACPO Good Practice Guide ACPO Good Practice Guide for Digital Evidence," March 2012. [Online]. Available: https://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf. [Accessed 4 April 2018].

[7] S. W. G. o. D. Evidence, "SWGDE," 12 April 2006. [Online]. Available: https://www.swgde.org/documents/Archived%20Documents/SWGDE%20Data%20Integrity%20Within%20Computer%20Forensics%20V1-0. [Accessed 18 06 2019].

[8] K.-s. L. C. C. Y. W. Feng Liu, "Research on the technology of iOS jailbreak," in *Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control*, Hefei, China, 2016.

[9] Packt, "iOS boot process and operating modes," [Online]. Available: https://subscription.packtpub.com/book/networking_and_servers/9781783553518/3/ch03lvl1sec23/ios-boot-process-and-operating-modes. [Accessed 20 November 2018].

[10] InteliPaat, "iOS Architecture," [Online]. Available: https://intellipaat.com/tutorial/ios-tutorial/ios-architecture/. [Accessed May 2019].

[11] S. Bhardwaj, "Core OS Layer in iPhone," 14 March 2013. [Online]. Available: https://www.c-sharpcorner.com/UploadFile/d49768/core-os-layer-in-iphone/. [Accessed 27 May 2019].

[12] J. Levin, Mac OS X and iOS Internals, Indianapolis: John Wiley & Sons, Inc., 2013.

[13] J. Zdziarski, "iOS Forensic Investigative Methods," May 2013. [Online]. Available: https://www.zdziarski.com/blog/wp-content/uploads/2013/05/iOS-Forensic-Investigative-Methods.pdf. [Accessed 22 January 2019].

[14] Apple, "Apple File System Reference," 7 February 2019. [Online]. Available: https://developer.apple.com/support/apple-file-system/Apple-File-System-Reference.pdf. [Accessed 18 March 2019].

[15] Scar, "Jailbreaking iOS 11 And All Versions Of iOS 10," 30 March 2018. [Online]. Available: https://articles.forensicfocus.com/2018/03/30/jailbreaking-ios-11-and-all-versions-of-ios-10/. [Accessed 28 December 2018].

[16] Electronic Frontier Foundation, "Unintended Consequences: Fifteen Years under the DMCA," March 2013. [Online]. Available: https://www.eff.org/id/pages/unintended-consequences-fifteen-years-under-dmca. [Accessed 5 November 2018].

[17] B. Azad, "Affect voucher_swap to Data Integrity," in *email*, 2019.