Poster: iSecureRing: Forensic ready Secure iOS apps for jailbroken iPhones

Jayaprakash Govindaraj
IIIT Delhi, New Delhi, India
jayaprakashg@iiitd.ac.in

Rashmi Ainahalli Mata
Infosys Labs, Bangalore, India
Rashmi_A02@infosys.com

Robin Verma
IIIT Delhi, New Delhi, India
robinv@iiitd.ac.in

Gaurav Gupta
IIIT Delhi, New Delhi, Indian
gauravg@iiitd.ac.in

***Abstract*—Apple's iOS is one of the major players in the smartphone market and it restricts installation of apps which are not from Apple app store. Users often resort to jailbreak their iPhones to break free from these restrictions. Considering jailbreaking iPhones is legal in the US; more devices are expected to be jailbroken in future. Jailbroken iPhones are making their way into enterprises, which allow Bring Your Own Device (BYOD), but these devices are either barred or restricted by Mobile Device Management (MDM) softwares which consider them as a security risk. In this work, we have designed a solution (iSecureRing) to secure mobile apps and to preserve date and time stamps of events to handle any security incidents in the jailbroken iPhones. To the best of our knowledge, iSecureRing is the first forensic ready mobile app security solution to secure an application running in an unsecure environment within the enterprise environment.**

I. INTRODUCTION

According to a report by Pew Research [1] 44.64% of the adult American population owning smartphone use the Apple's iPhone. Apple's iOS does not allow installation of additional applications, extensions and themes that are not available through Apple's App store. Users jailbreak (process to get root level access) their devices to break away from these restrictions [2]. Once jailbroken, iPhone allows retrieval of applications and corresponding data stored on it, hence compromising the security of the applications and confidentiality of data [3]. As per US copyright office, jailbreaking of iPhone continues to be legal [2] [5]. Considering jailbreaking to be a reality, there is a need to design ways to secure mobile applications running even in the jailbroken iPhones. This requirement of making an application secure in an unsecure environment is critical to the enterprise environment where proprietary application(s) should work without impacting the enterprise security. Currently, enterprises allowing BYOD generally detects and restrict jailbroken iPhones with MDM softwares, like Citrix's XenMobile, IBM's Endpoint manager etc. Employees have to either un-jailbreak their iPhones or use another device to install the enterprise application(s). With our solution the enterprises can install their application securely on employee's jailbroken iPhone. Any new or existing apps can be secured and made forensics friendly, even if the iPhone is jailbroken.

II. IMPLEMENTATION METHODOLOGY

Our solution consists of two modules; the first module consists of a static library that can wrap apps in an additional layer of protection making them difficult to crack on jailbroken devices and thus preventing access to application data. Second module preserves authentic date and times stamps of the events related to the secured app, so that in case of any security incident digital forensic analysis can be performed [6]. The captured timestamps are stored outside the device on a secure server or the cloud. The modules are discussed in detail in following subsections.

A. Securing the Apps

This static library consists of various APIs that can be used to identify security vulnerabilities in jailbroken iPhones. This library can be used to detect and mitigate security issues. The library mainly contains functions namely, `isCheck1()`– iPhone is jailbroken or not; `isCheck2()` – application is running in debug mode or not; `enableDB()` – For disabling the gdb (debugger) for a particular application (process); `isAppC()` – If application's binary is still encrypted and also check for application bundle files (Info.Plist) integrity; `initialize()` – If any of the static library function themselves are hooked or not; `CheckA()` – critical methods (functions) passed as an argument is hooked or not; `CheckS()` – any methods/functions related to SSL Certificate Validation are hooked or not; `CRCCheck()` – To find if the application is tampered or not.

B. Preserving the date and time stamps

We have created a Dynamic library using *MobileSubstrate* framework, this framework provides APIs to add runtime patches or hook to the system functions on jailbroken iOS [7]. The solution architecture (refer *Fig. 1*) consists of four components:

Dynamic library – hooks on to the system open calls and captures kernel level date and timestamps corresponding to selected file(s) and then writes them to the log file.

Timestamp log file – the log file is stored in the internal memory of the iPhone which is not directly accessible to applications making it safe against deletion attempts.

Uploading the log file – the log file generated by the DLL, is later uploaded at regular intervals to an external server or cloud based on the network connectivity.

External server/cloud – external server within the enterprise environment or on a secure location on the cloud

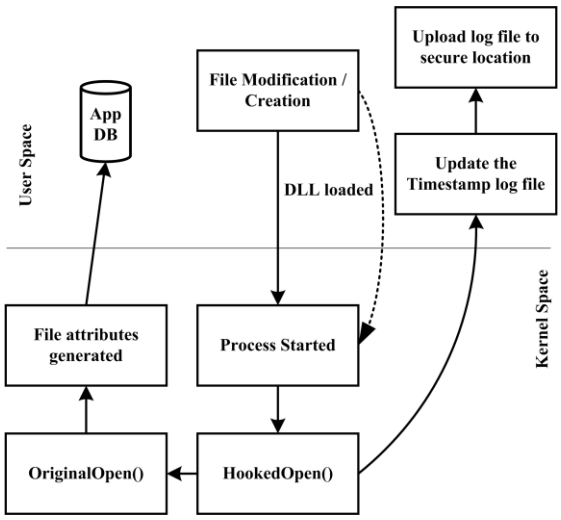

Fig. 1. Solution architecture preserving date and timestamps

This Dynamic library (dylib) will be loaded into all the running applications. Filters can be applied to this dylib so that it gets loaded only for specified applications. These filters are implemented as Property-List file that is added along with dylib into jailbroken iPhone. The filter should be a dictionary with main key filter and other keys bundles, classes, executables. Ex: - Filter = { Executables = {"mediaserverd"}; Bundles = {com.apple.mobileslideshow"}; };

III. EXPERIMENTS AND RESULTS

We created two apps one without any protection and another using iSecureRing and deployed them on a jailbroken iPhone 4 (iOS 7.0.6). We simulated a series of attacks on the Apps and the data to validate our solution. At the application level, the Apps were subjected to various attacks to exploit lack of binary protection vulnerability [8] as illustrated in the Table 1.

TABLE I. ATTACKS AND RESULTS

iPhone 4 (iOS 7.0.6)	Jail breaking	Debug mode	Encryption check	Hooking	Code Tampering
Non Jailbroken (*App with no protection*)	✓	×	×	×	×
Jail broken (*App with no protection*)	NA	✓	To be Done	✓	✓
Jail broken (*App with iSecureRing*)	NA	×	×	×	×

The results demonstrate that the App with iSecureRing on a jail broken iPhone (Row 3, Table 1) is as secure as an normal App running in a non-jail broken iPhone (Row 1, Table 1).

The iSecureRing also helps in detecting any attempts made to exploit known or unknown vulnerabilities by capturing the timestamps of activities associated with the secured app. We simulated a timestamp tampering attempt on one of the images from Apple's Photo app. iSecureRing successfully captures all the events in the log. Using the log we were able to identify the tampering attempts. *Fig. 2* illustrates the MAC DTS (Modified Accessed Created Date and Time stamps) [6] captured for one of the images.

```
file=/System/Library/PrivateFrameworks/TextInput.framework/Info.plist
Mtime=Wed Feb 19 11:10:02 2014
  Atime=Wed Feb 19 11:10:02 2014
  Ctime=Wed Feb 19 11:10:02 2014
  Flag=0

file=/var/mobile/Media/PhotoData/Thumbnails/V2/DCIM/100APPLE/IMG_0002.PNG/5003.JPG
Mtime=Sun Mar  2 23:27:24 2014
  Atime=Sun Mar  2 23:27:24 2014
  Ctime=Sun Mar  2 23:27:24 2014
  Flag=0
```

Fig. 2. MAC DTS logs

We conducted performance benchmarking for the 3 cases considered in our experiment, *Fig. 3* summarizes the results from our initial tests (5 runs). The results show that no significant difference in the performance of the device.

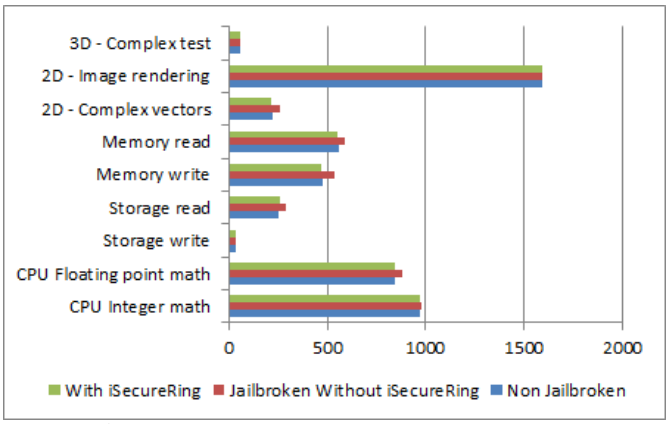

Fig. 3. Performance benchmark results

IV. CONCLUSION AND FUTURE WORK

We demonstrated that iSecureRing can be deployed to make a secure and forensic ready App in unsecure jailbroken iPhone in a enterprise BYOD environment. This solution can be further extended to other mobile devices OS like android, windows phone etc.

REFERENCES

[1] Aaron Smith, Smartphone ownership–2013 update. Pew Research Center: Washington DC, 2013.

[2] Andrew Hoog and Katie Strzempka. iPhone and iOS Forensics: Investigation, Analysis and Mobile Security for Apple iPhone, iPad and iOS Devices. Elsevier, pp. 14-15, 2011.

[3] Charlie Miller, Mobile attacks and defense. Security & Privacy, IEEE 9.4 (2011): 68-70.

[4] Andy Greenberg, Evasi0n Is The Most Popular Jailbreak Ever: Nearly Seven Million iOS Devices Hacked In Four Days, online article at forbes.com, last accessed on 31st March, 2014 at http://goo.gl/PDGJL

[5] Sean Morrissey and Tony Campbell. iOS forensic analysis for iPhone, iPad, and iPod touch. Vol. 23. Apress, pp. 271-272, 2010.

[6] Robin Verma, Jayaprakash Govindaraj, and Gaurav Gupta. Preserving date and timestamps for incident handling in android operating system. Proceedings of Tenth Annual IFIP WG 11.9 International Conference on Digital Forensics, 2014, in press.

[7] Mathieu RENARD. Practical iOS Apps hacking. G 2 reHack 012: 14.

[8] Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra. A survey on security for mobile devices. Communications Surveys & Tutorials, IEEE 15, no. 1 (2013): 446-471.