Privacy and Security Benefits of Jailbreaking iOS

Stefan Dimitrov
sdimit01@tufts.edu

Ming Chow
Mentor

December 10, 2013

Abstract

The traditional software distribution channel on the iOS platform is Apple's own AppStore, which is known for its stringent app approval policies and quality control. One of the aims of Apple's approval process is to prevent malicious software from reaching its customer base. This preemptive strategy has proven to be relatively successful, but has been criticized for rejecting benign apps that replace or enhance core services. Jailbreaking has opened an alternative outlet for this niche of software offerings by letting users install apps from third party software repositories outside of Apple's regulation, however, this lack of security audit theoretically enables the unhindered distribution of potentially malicious code. Moreover, because jailbreaks exploit a security vulnerabilities within iOS in order to grant apps root privileges and unrestricted filesystem access, they effectively disable several security features in iOS. However, jailbreaking also allows the installation of third-party patches targeting those very vulnerabilities. In addition, jailbreak tweaks that improve privacy control have surfaced in response to controversial AppStore apps that misuse user information, yet pass the approval process. These subtle, but significant differences between the stock and jailbroken flavors of iOS motivate this paper in an attempt to explore the pros and cons of jailbreaking regarding security and privacy.

.

Contents

1 Introduction

The App Store is an integral part of the iOS security model. All applications on the App Store must go through an approval process, one of whose purposes is to prevent malicious apps from reaching users. This process involves quality control, static and manual analysis of the behavior and content of apps. In addition the use of private APIs, which typically would allow an app to provide unapproved or malicious functionality is screened against. Apps on the iOS platform are assigned a unique identifier and a folder whose contents it can modify, while the rest of the filesystem is out of reach. Individual apps can communicate with other apps in a very limited fashion by means of URL Handlers. Access to Apple services such as Push Notifications, Game Center and Passbook is authenticated through certificates issued by Apple. Moreover, every application on the App Store is code signed, and code that hasn't been signed by Apple cannot run on the device. All these measures ensure that apps run in a sandboxed environment which ensures that the system and the user data on the device are protected in case an app attempts malicious activity.

Jailbreaking is the process of modifying iOS through what is most often a security exploit in order to allow code that hasn't been signed by Apple to run on the device, in addition to enabling root access to the filesystem for any app. Hence jailbreaking allows for downloading apps from sources exempt of Apple's stringent approval procedures, which typically enhance the functionality of the OS and its stock apps and allow for unapproved use of private API that provide greater functionality and freedom to developers.

Below I outline researchers' findings on circumventing the approval process and

examples of malicious apps on the App Store, in order to expose the reliance on the walled garden model as insufficient with regards to ensuring security and privacy.

2 To the Community

This paper is aimed at iOS researchers and advanced users, and seeks to dispel the notion that jailbreaking is fundamentally insecure. With this paper I hope to raise awareness about the dangerous sense of false security and privacy that comes with Apple's walled garden model of software distribution.

Jailbreaking can be invaluable when it comes to security and privacy research on the iOS platform. Unrestricted access to the platform's internals and filesystem enables insight into security and privacy strengths and weaknesses, given that iOS is closed source. Moreover, jailbreaking allows researchers to run code unsigned by Apple, use Apple's private frameworks and even patch the OS itself, which can aid in discovering system vulnerabilities and developing fixes for them.

Privacy and security enhancing software offerings in their essence must be able to modify or access low level system functionality in order to detect apps leaking information or misusing privileges and prevent this from happening. Since apps that access the system internals and private frameworks are strictly forbidden, such apps have only been available to the users of jailbroken iOS devices.

Considering these important uses of jailbreaking, in this paper I present third-party research that benefits from jailbreaking in evaluating the state of security and privacy on iOS, and showcase useful tools that can improve the security and privacy

of jailbroken iOS users.

3 Sneaking malicious code into the AppStore

Prior to submitting an app each developer needs to enroll into the paid Apple Developer Program and disclose their identity. This allows Apple to ban developers caught committing infractions from submitting apps to the App Store. While this might discourage developers of malicious apps under the threat of receiving a permanent ban from the App Store, it would be unwise to believe in the benevolence of developers solely due to the existence of a disincentive to misbehaving.

In reality, since the approval process does not require that developers submit their source code for a review, it is quite possible to use code obfuscation techniques in order to evade any static or dynamic code analysis tools Apple might be using to uncover threats. In his paper on iOS privacy[8], Nicolas Seriot demonstrates several simple techniques that can be used to deliberately circumvent common types of code analysis likely used by Apple in search of calls to private APIs and system directories. He suggests that his proof-of-concept SpyPhone app[1] in fact does not rely on private APIs in the first place, and goes on to express concern that the existence of apps on the App Store collecting personal data is quite likely given the underground demand for and abundance of personal data on iOS devices.

Another group of researchers from Georgia Institute of Technology present[10] their "Jekyll App" which obtained Apple's approval, despite allowing the researchers to remotely launch a variety of attacks on their own tests devices. The paper demon-

[1] https://github.com/nst/spyphone/

strates in practice how it is possible to create malicious apps that evade detection by breaking down malicious code into distinct subroutines termed "gadgets" which are not malicious on their own but if *remotely* supplied the correct execution paths can be assembled into a malicious routine. Breaking down code with malicious effects into innocuous looking subroutines practically renders static analysis less useful as the malicious code no longer appears explicitly. Moreover, since the malicious routines are enabled remotely, even the effectiveness of manual testing and dynamic analysis is diminished.

[5],

4 Threats to Privacy in the AppStore

Even though researchers have demonstrated techniques such as dynamic [9] and static [2] analysis targeted at capturing data flows within the application have been developed, several controversial cases of user data misuse and user profiling have gained attention[2].

A recent case of privacy violations emerged in November 2013, as the developer Kyle Richter discovered that a popular game app QuizUp transmitted sensitive information about a players Facebook friends, Address Book contacts and others, in a plain text format over to the company servers, presumably in an effort to match the user with other players.[3] Despite the fact that thanks to the privacy enhancements in iOS, the app had to request users' permission to collect the data, the users are still

[2]http://mclov.in/2012/02/08/path-uploads-your-entire-address-book-to-their-servers.html
[3]http://kylerichter.com/our-responsibility-as-developers/

not able to fine tune the permissions granted, for example by disallowing access to contacts' email addresses or real names, which wouldn't be of use to a game matching algorithm.

While AppStore apps can pose threats to privacy, iOS itself collects large amounts of data from normal usage. In 2011, a controversy arose due to iOS maintaining a database of WiFi hotspots and cell towers around the users location, in order to improve its geo-location services beyond complete reliance on the slower GPS systems which would allow for quickly determining the user's current location when using Maps.[4] Even though, Apple assured that despite this location data being sent to Apple servers, it is completely anonymous the data would remain on the device itself in a file named consolidated.db which can then be easily retrieved.[5].

A new feature in iOS7, called Frequent Locations relies on a similar location data storage, which contains the users most frequently visited locations. The idea behind the feature is to enable Next Destination, a user-end feature that lets Apple provide relevant commute information such as travel time to work.[6] Despite users being able to disable those features, they are enabled by default and the data collection is not apparent.

[4]http://www.networkworld.com/community/blog/apple-officially-responds-ios-consolidateddb-
[5]https://www.sans.org/reading-room/whitepapers/forensics/forensic-analysis-ios-devices-34092
[6]http://www.idownloadblog.com/2013/10/20/how-to-use-frequent-locations-and-next-destination-features-in-ios-7/

5 Threats of Jailbreaking

you can crack the pin code of someone by jailbreaking an un jailbroken phone first and then changing the pin you can use the espionage tools in [7] unless you have stuff fixed. default root password is very well known. can't update if u want to keep the jailbreak. installing malicious apps from evil repositories.

6 Action Items

6.1 Jailbreaking for better privacy

Before address book permissions were first added to iOS 6, the well publicized case of the social networking app Path drew attention to the issue of privacy in the AppStore. It was discovered that Path collected personal information from the user's address book and sent it to company servers without the users consent and it did not list this activity in the privacy agreement. The creators of Path have since removed the behavior and settled the ensuing FTC charges. [7] In response to this, a tweak for jailbroken iOS devices named ContactPrivacy appeared. It allowed the user to control access to the address book.

6.2 Changing the default iOS root password

Users with jailbroken devices are advised to change the default root password - the well known "alpine". Since an attacker can install a remote login utility such as

[7] http://www.ftc.gov/news-events/press-releases/2013/02/path-social-networking-app-settles-ftc-charges-it-deceived

SSH and gain full control over the device. In case a device is jailbroken by an attacker, the default password if left unchanged can be used to access the filesystem, bypassing any lock screen passwords the user might have (passwords are stored in `\var\Keychain`)[6]

6.3 Jailbreaking for security and privacy research

In order to perform any sort of dynamic analysis on a device, root access or at least filesystem access is necessary. Apple does not grant those on unmodified iOS devices and even developers are not granted direct access to the filesystem of the device. Jailbreaking is useful here as it provides researchers with unrestricted access to the OS. A group of researchers from Vienna University of Technology and UC Santa Barbara[9], developed a method for dynamic analysis that relies on a VNC server for the automation of UI manipulation which is necessary to make sure that all code is being run while testing. Since the App Store restrictions pose technical limitations to implementing VNC servers, only the dynamic analysis method developed can only be used on jailbroken devices. Nevertheless, even more traditional tools such as `gdb` cannot be installed on iOS unless a jailbreak is in place.

One such vulnerability[8] in the PDF rendering engine of Apple iOS[2], was successfully used to jailbreak an earlier version of iOS by allowing the code necessary to obtain elevated privileges run on the device.

[8] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-0226

7 Conclusion

The problem of course is somewhat intractable, because trusting data to a third party comes with no guarantees about how the data will be used. Nevertheless, it is important to make sure that users are given the power to *decide* whether to give their data in the first place. This privacy enhancement feature can be a great complement to pre-existent code approval policies and jailbreaking has allowed for several such data flow management tools. However it wasn't until iOS7 before users were able to control microphone use or limit ad tracking[9] and a dedicated Privacy Settings configuration pane providing consolidated control over the permissions of individual apps has been available only as of iOS6[10].

[9] http://blogs.wsj.com/digits/2013/09/18/how-to-use-apples-new-ios-7-privacy-controls/
[10] https://developer.apple.com/library/ios/releasenotes/General/RN-iOSSDK-6_0/

References

[1] Dimitrios Damopoulos, Georgios Kambourakis, and Stefanos Gritzalis. iSAM: An iPhone Stealth Airborne Malware. In Jan Camenisch, Simone Fischer-Hübner, Yuko Murayama, Armand Portmann, and Carlos Rieder, editors, *Future Challenges in Security and Privacy for Academia and Industry*, volume 354 of *IFIP Advances in Information and Communication Technology*, pages 17–28. Springer Berlin Heidelberg, 2011.

[2] Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. PiOS: Detecting Privacy Leaks in iOS Applications. In *NDSS 2011, 18th Annual Network and Distributed System Security Symposium, 6-9 February 2011, San Diego, CA, USA*, San Diego, USA, 02 2011.

[3] Stefan Esser. Exploiting the iOS Kernel. In *Black Hat USA*, 2011.

[4] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. A Survey of Mobile Malware in the Wild. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, pages 3–14, New York, NY, USA, 2011. ACM.

[5] Jin Han, SuMon Kywe, Qiang Yan, Feng Bao, Robert Deng, Debin Gao, Yingjiu Li, and Jianying Zhou. Launching Generic Attacks on iOS with Approved Third-Party Applications. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*,

volume 7954 of *Lecture Notes in Computer Science*, pages 272–289. Springer Berlin Heidelberg, 2013.

[6] Matthias Boll Jens Heider. Lost iPhone? Lost Passwords. *Fraunhofer Institute for Secure Information Technology (SIT)*, 2011.

[7] Keith Lee. iPhone Espionage. *http://reverse.put.as/wp-content/uploads/2011/06/D2-SIGINT-Keith-Lee-iPhone-Espionage.pdf*, 2011.

[8] Nicolas Seriot. iPhone Privacy. In *Black Hat DC*, 2010.

[9] Martin Szydlowski, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. Challenges for Dynamic Analysis of iOS Applications. In Jan Camenisch and Dogan Kesdogan, editors, *Open Problems in Network Security*, volume 7039 of *Lecture Notes in Computer Science*, pages 65–77. Springer Berlin Heidelberg, 2012.

[10] Tielei Wang, Kangjie Lu, Long Lu, Simon Chung, and Wenke Lee. Jekyll on iOS: When Benign Apps Become Evil. In *USENIX Security '13*, 2013.

[11] Tim Werthmann, Ralf Hund, Lucas Davi, Ahmad-Reza Sadeghi, and Thorsten Holz. PSiOS: Bring Your Own Privacy & Security to iOS Devices (Distinguished Paper Award). In *8th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2013)*, May 2013. Distinguished Paper Award.